# Characterizing and Overcoming the Limitations of Neural Autoregressive Models

Kartik Goyal

CMU-LTI-21-005

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

**Thesis Committee:**
Chris Dyer *(chair)*
Taylor Berg-Kirkpatrick *(chair)*
Graham Neubig
Alexander Rush

*Submitted in partial fulfillment of the requirements*
*for the degree of Doctor of Philosophy*
*in Language and Information Technologies.*

# Abstract

Neural sequence models are typically parametrized as autoregressive models that are locally normalized. These models simplify the generation process by generating constituent tokens in a predetermined order in a stepwise manner guided by a probability distribution over the vocabulary of tokens at each step. Although they have achieved impressive performance on several language processing and generation tasks like machine translation, dialog response generation, speech processing and synthesis etc., this class of models is also known to exhibit degenerate behavior during optimization and decoding. In this thesis, I characterize some of the limitations of locally normalized models, namely exposure bias and label bias, both of which represent pernicious inductive biases associated with autoregressive models that preclude efficient training of such deep neural models. This dissertation proposes solutions to ameliorate such issues in order to train more powerful and well-behaved probabilistic sequence models.

To ameliorate *exposure bias*, this thesis presents two solutions that focus on making the training of the models more aware of the behavior of the downstream decoding algorithms for proper credit assignment for digression from the reference sequence during gradient-based training. The presented solutions crucially involve continuous relaxations to the commonly used discontinuous decoding procedures with neural sequence models including greedy arg-max decoding, ancestral sampling, and beam search, to enable gradient-based optimization using automatic differentiation libraries . These approaches are empirically superior to standard approaches for various natural language processing tasks like machine translation, CCG supertagging, and named entity recognition.

Next, this dissertation focuses on an entirely new class of probabilistic sequence models–globally normalized models–that accommodates more flexible generation procedures and is unlikely to suffer from exposure bias and label bias but involves tradeoffs in computational complexity. A method to train globally normalized sequence models is introduced which involves modification of the above-mentioned search-aware algorithm involving the continuous relaxation of beam search. The empirical comparison of such globally normalized models with their locally normalized counterparts, also trained via the continuous relaxation to beam search reveals that training with the globally normalized strategy results in models that are more effective at responding to search errors during training.

Following this promising behavior of globally normalized models, this thesis explores the energy-based modelling view of fully connected globally normalized models and proposes powerful bidirectional energy parametrizations for sequences. Specifically, this thesis interprets optimization of popular *masked language models* (MLMs) as implicit training of energy-based sequence models and introduces a strategy to correctly sample from MLMs that do not have a probabilistic interpretation on their own. This work not only introduces a strategy to sample from the MLMs but also provides evidence for efficient indirect training of energy-based sequence models.

To conclude, while autoregressive models are easy to train and efficient to use, they

are addled by poor inductive bias and exhibit degenerate behavior. While the alternative class of globally normalized models comes with limitations around computational complexity, it offers amenability toward more flexible and powerful sequence models.

# Acknowledgments

A great many people have influenced my thoughts, life and this thesis over the past few years. I am confident that despite my best efforts, I would fall short of acknowledging all of these people's impact, but here is my attempt anyway.

First and foremost, I would like to thank my advisors Chris Dyer and Taylor Berg-Kirkpatrick for their constant support and guidance. I am extremely grateful to them for providing me encouragement to explore my research interests while remaining ever available to discuss topics in great philosophical and technical depth with amazing clarity of thought. I would also like to thank my committee members Graham Neubig and Alexander Rush for the discussions and suggestions that have influenced the contents of this document.

I would like to thank all the people who have worked with me on research projects and from whom I have learned so much: Nikolai Vogler, Christopher Warren, Max G'Sell, Samuel Lemley, Shruti Rijhwani, Pierce Williams, Avery Wiscomb, Alon Lavie, Lori Levin, David Mortensen, Eduard Hovy, David Rosenberg, Gideon Mann, Austin Matthews, Greg Hanneman, Michael Denkowski, Patrick Littell, Alexa Little, Sam Thomson, Swabha Swayamdipta, Sujay Jauhar, Mrinmaya Sachan, Shashank Srivastava, Dirk Hovy, Huiying Li. I am also extremely grateful to friends and colleagues I have met at CMU for their insightful conversations and patience with my ramblings: Willie Neiswanger, Maria Ryskina, Nikita Srivatsan, Arnav Kumar, Anjalie Field, Michael Miller Yoder, Qinlan Shen, Avinava Dubey, Snigdha Chaturvedi, Volkan Cirik, Daniel Spokoyny, James Route, Liz Salesky, Manaal Faruqui, Torsten Wortwein, Pradeep Dasigi, Yulia Tsvetkov, Waleed Ammar, and Jonathan Barker. I would also like to thank the staff and professors at Language Technologies Institute who keep the PhD academic program running smoothly.

I would also like to thank all of the amazing and inspiring friends I have made in Pittsburgh over the years who have been instrumental in making my time here extremely enjoyable. In particular, I would like to thank my climbing friends with whom I have shared a number of memorable adventures in the wilderness: Michael B, Aria, Derek, Michael C, Nico, Nate, Emily, Chris, Colin, Ollie, Ali, Simon, Ray, Joe, Fine, and Ramsey. I would also like to thank my boardgame friends–Alan and Yubin, my GroupX friends, and so many other wonderful bouldering friends at Ascend who have kept the *psych* high.

Finally, I would like to give my deepest thanks to my parents and my sister Bharti who have been a constant supportive presence throughout my life.

# Chapter 1

# Introduction

Neural sequence models have been successful at probabilistic-modeling of rich and structured outputs associated with combinatorial objects likes text documents which, in turn have been used for several important tasks like machine translation, summarization etc. (Sutskever et al., 2014; Rush et al., 2015). These neural sequence models are overwhelmingly operationalized as *autoregressive locally-normalized* models which leads to easy and computationally efficient training and decoding via the maximum likelihood principle. These autoregressive models are the state-of-the-art models for processing and generation of sequences and have become a cornerstone for applications like automatic translation, speech synthesis, dialog generation, image synthesis etc. At their core, locally normalized models break down the problem of generating a sequence into sequential generation of smaller constituent chunks, most commonly tokens, in a predetermined order (left-to-right by default) wherein each generation step is specified by a probability distribution over the vocabulary conditioned on the previously generated tokens. Chain rule ensures easy computation of probability of sequences using these conditional distributions which leads to ease of optimization and usage. Despite their ubiquity, this class of models is known to exhibit degenerate behavior (Koehn and Knowles, 2017; Stahlberg and Byrne, 2019; Wang and Sennrich, 2020) during optimization and decoding. In this thesis, I characterize some of the limitations of locally normalized models and

propose solutions to ameliorate such issues in order to train more powerful and well-behaved probabilistic models for sequences. While a set of solutions described in this thesis focus on improving the training of these locally normalized autoregressive models, this thesis eventually explores and develops *globally normalized models*, which come with their own computational trade-offs, as alternative to autoregressive models to ameliorate the degenerate behavior associated with the autoregressive models.

Specifically, this thesis identifies two major issues associated with optimization and operationalization of autoregressive locally normalized models:

- Exposure Bias: Maximum likelihood optimization via *teacher-forcing* algorithm causes discrepancy between training and decoding with autoregressive locally normalized models. Essentially, the sequential neural models encounter novel continuous states in the representation space of the context during decoding which were never encountered during training of these models which leads to degenerate output from these models.
- Label Bias: The normalization constraint over vocabulary items at each decoding step in autoregressive models poses a harmful inductive bias which leads to learning miscalibrated distributions over tokens and sequences. In this thesis, I refer to this undesirable inductive bias by *label bias* [1] which leads to several negative consequences like inability to recover from wrong decisions made in the earlier decoding steps, and ignoring the input sequence in case of conditional generation and overtly focusing on the decoded token history.

## 1.1 Ameliorating exposure bias

To ameliorate *exposure bias*, the work in this thesis presents two solutions that focus on making the training of the models more *search-aware* with the general idea being that of informing the training

[1] While this is related to the notion of label bias explored by McCallum et al. (2000) and Bottou (1991), I use the term to refer to a broader set of consequences as a result of the harmful inductive bias associated with the normalization constraint in autoregressive models.

procedure of the behavior of the decoding algorithms to be used during deployment. This causes the optimization objective to also include the ability to recover from digressions in the step-wise generation of sequences. One major bottleneck to achieve this efficiently is that the decoding algorithms are a *discontinuous* function of the model parameters and hence cannot be associated with gradients which is crucial for optimization of neural networks using automatic differentiation. Therefore, the solutions introduced in this document involve continuous relaxations to the commonly used discontinuous decoding procedures with neural sequence models. One of the solutions (Goyal et al., 2017) (described in **Chapter 3**) is inspired from the *scheduled sampling* (Bengio et al., 2015) procedure and extends it by relaxing the procedure via temperature-based softmax such that the objective is differentiable and aware of discontinuous greedy decoding or sampling performed during inference. Another solution (Goyal et al., 2018) (described in **Chapter 4**) focuses on making the training procedure aware of beam search as the inference procedure of choice during decoding. This method introduces a continuous relaxation of the beam search procedure so that the effect of beam search can be incorporated directly in the computational graph.

The continuous relaxation of the discontinuous decoding procedures results in successful end-to-end backpropagation-based training which requires the presence of gradients or subgradients. We empirically demonstrate the effectiveness of these approaches via significant improvements over teacher-forcing for tasks like Machine Translation, Named Entity Recognition, and CCG supertagging.

## 1.2 Globally normalized models for addressing label bias

While the problem of exposure bias can be addressed in a targeted manner by the approaches described above, this thesis explores a different class of sequence models from the autoregressive locally normalized models – globally normalized models – which involve exposure to all of the negative space of sequences via a computationally expensive partition function. Thus, in spite of

being computationally expensive and difficult to optimize and decode from, this class of models is unlikely to suffer from exposure bias in the first place. Therefore in **chapter 5**, a method to train *globally normalized* sequence models is introduced (Goyal et al., 2019) in which we modify the above-mentioned search-aware algorithm involving continuous relaxation of the beam search. The comparison of such globally normalized models with their locally normalized counterparts, also trained via the continuous relaxation to beam search reveals that in the context of inexact inference using beam search, training with the globally normalized strategy results in models that are more effective at responding to search errors during training compared to their locally normalized counterparts. This is in spite of the fact that assuming flexible neural parametrization, locally normalized models are theoretically as expressive as globally normalized models.[2] This finding is attributed to globally normalized models ameliorating the issue of *label bias* commonly associated with locally normalized models. In this work, we use the lens of *label bias* to explain the potential reason for amenability of globally normalized sequence models to search-aware training in context of inexact search.

## 1.3 Masked language models and globally normalized models

Encouraged by the evidence that globally normalized models are less susceptible to exposure bias and ameliorate label bias, this thesis explores more expressive globally normalized models for sequences that construct a scalar score for the sequences in a stepwise manner in which each step uses the full bidirectional context to compute the components contributing to the sequence score. This strategy views training of *globally normalized* models as minimization of *energy* in an energy-based interpretation of the globally normalized models where the energy for a sequence directly corresponds to its scalar score. Training such energy based models is difficult because it typically involves approximating an intractable partition function over the space of sequences. As an

[2]The work in this thesis assumes sequences are limited to a finite maximum length, which implies that the search space over sequences is also finite and countable.

alternative to training such energy-based sequence models directly, in **Chapter 6** this thesis (Goyal et al., 2021) interprets optimization via the widely popular *masked language modeling* (MLM) objective as implicit training of energy-based sequence models and introduces a strategy to correctly sample from the masked language models that do not have a probabilistic interpretation on their own. Pretrained MLMs abandon the chain rule and left-to-right order, but instead use bidirectional context to construct *representations* for the tokens and sequences. These representations have shown to be extremely useful for various natural language processing (NLP) tasks. Hence, MLMs are powerful *encoders* but they are *not probabilistic generators of language*. This thesis proposes two energy parametrizations that can be easily computed with pretrained MLMs and introduces an effective Metropolis-Hastings based sampler that uses the MLM free conditionals as Monte Carlo transition distributions for proposals in the MCMC chain to generate high quality samples from the pretrained MLMs. This work not only introduces a strategy to sample from the ubiquitous MLMs but also shows empirically, on the conditional generation task of machine translation, and theoretically that MLM objectives result in training of an implicit energy model over sequences.

While it is inspiring that the masked language modelling objective easily trains effective energy based models over sequences which suggests that other efficient optimization procedures could be developed to train powerful and expressive globally normalized models with desired inductive biases; in general, the computationally expensive partition function makes it difficult to train energy-based model for sequences. Not only the optimization, but sampling from the energy based models also is more expensive than sampling from autoregressive models. For example, running Monte Carlo Markov Chain as described in Chapter 6 is much more expensive than performing ancestral sampling via autoregressive models. Therefore, while globally normalized models are better at incorporating inductive bias to ameliorate issues associated with autoregressive models and provide more explicit control over the global constraints and features for controlled generation of sequences, they come at a computational cost of estimating or approximating a partition function that grows exponentially with the length of sequences considered.

This dissertation is organized in the following manner. In **Chapter 2**, general mathematical notation that is used throughout this document is described and background for autoregressive and globally normalized models is established. **Chapter 3** focuses on a technique based on scheduled sampling to ameliorate exposure bias and **Chapter 4** introduces continuous relaxation to the discontinuous beam search procedure for search-aware optimization of locally normalized models to counter exposure bias. **Chapter 5** introduces and explores an approach to train globally normalized sequence models by the aforementioned relaxation to beam search. The importance of treating globally normalized models as energy based sequence models is described in **Chapter 6** with a bulk of this chapter focusing on exploring the relationship between energy based sequence models and recently popular and effective masked language modeling objective. Finally, I conclude with some closing thoughts in **Chapter 7**.

# Chapter 2

# Background

## 2.1 Neural sequence models: Notation

In this document, the term *neural sequence models* refers to the kinds of models that are parametrized via neural networks and that output (or generate) a sequence of discrete symbols denoted by $\mathbf{y}$, unless otherwise noted. We refer to the module containing parameters for generating these symbols by the term neural *decoder*. The generation of these symbols could be unconditional, or it could condition on some other input $\mathbf{x}$. In case of conditional generation, another neural module, called an *encoder*, parametrized differently from the decoder *processes* the input $\mathbf{x}$, and learns a vector-based representation that is fed as input to the decoding process for generation of the output $\mathbf{y}$. All the work in this thesis focuses heavily on the *decoder* that generates the output sequence $\mathbf{y}$, and the claims, results, and analysis are largely agnostic to the type of encoder in the sequence model. Unless otherwise noted, the applications considered in this thesis to perform empirical comparison involve conditional generation where the output $\mathbf{y}$ is conditioned on an input sequence $\mathbf{x}$. This is the case because of availability of well-defined and commonly accepted metrics for evaluation of prediction-based performance in these conditional generation applications. For all our analysis, we assume that the output $\mathbf{y}$ is limited to a certain maximum length so that generation of infinitely long

sequences is precluded, and the output space of all the possible outcome spaces, denoted by $\mathcal{Y}$, is finite and countable. In case data for supervised learning is available for above-mentioned sequence translation applications, the ground-truth sequence for the training objective is denoted by $\mathbf{y}$ . The length of a sequence is typically denoted by the symbol $T$. The individual tokens in a sequence are denoted by an index in the subscript (example $\mathbf{y}_t$ is the $t$-th token in $\mathbf{y}$) and collections of contiguous tokens are denoted by a range of indices (example, $\mathbf{y}_{t:s}$ refers to the span from $t$-th token to $s$-th token in $\mathbf{y}$) in the subscript. $\mathbf{y}_{<t}$ and $\mathbf{y}_{>t}$ refer to the spans in $\mathbf{y}$ preceding and succeeding the $t$-th token. $\mathbf{y}_{nt}$ refer to all the tokens in the sequence except the token at $t$-th position. At each position, generation of the output token is restricted to a set of discrete symbols called *Vocabulary*, denoted by $V$.

As described in the greater detail next, I refer to autoregressive models as being a class of models in which the output $\mathbf{y}$ is generated in step-wise manner with each step generating a token conditioned on only the previously generated tokens. Wlog. for left-to-right sequence order, the prediction at time step $t$ is only dependent on tokens $\mathbf{y}_{<t}$ and not on any succeeding ungenerated tokens. This work mainly deals with probabilistic autoregressive models which lend to computation of the probability of sequence easily via chain rule run left-to-right. It must be noted that this setup is agnostic to the parametrization of the decoder. In our description, the decoder could be parametrized as an RNN (Mikolov et al., 2010), LSTM (Hochreiter and Schmidhuber, 1997), transformer (Vaswani et al., 2017), or a GRU (Cho et al., 2014), and the arguments and the general claims made in this thesis are indifferent to the parametrization choice. The empirical comparison across this thesis is done with different decoding architectures in different chapters but the underlying issues, solutions, and general lessons and trends learned from the analysis are expected to hold across all such parametrizations.

## 2.2 Autoregressive Locally Normalized Sequence Models

For modeling sequences using neural networks, this class of models is the most prevalent in literature. As mentioned above, for autoregressive models, the probability of a sequence $\mathbf{y}$ is computed via chain rule run in a left-to-right manner where $\mathbf{y}_t$ is explicitly only dependent on the sequence history $\mathbf{y}_{<t}$. Under a locally normalized sequence model $\mathcal{M}_L$, the probability of $\mathbf{y}$ given $\mathbf{x}$ is:

$$p_{\mathcal{M}_L}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p(\mathbf{y}_t \mid \mathbf{x}; \mathbf{y}_{<t}) = \prod_{t=1}^{T} \frac{\phi_t(\mathbf{x}; \mathbf{y}_{<t}; \mathbf{y}_t)}{Z_{L;t}(\mathbf{x}; \mathbf{y}_{<t})}$$

where $Z_{L;t}(\mathbf{x}; \mathbf{y}_{<t}) = \sum_{y \in V} \phi_t(\mathbf{x}; \mathbf{y}_{<t}; y)$, is the local normalizer at each time step and $T$ is the number of prediction steps. The quantity $\phi_t(\mathbf{x}; \mathbf{y}_{<t}; y)$ is a positive scalar score for predicting the token $y$ at position $t$, whose computation is conditioned on the input sequence, and the output sequence history. For a typical autoregressive decoder, say an LSTM, the scalar score is computed as a logit that depends on the representation of the input sequence $\mathbf{x}$ computed by a separate sequence encoder, and the contextual representation which summarizes the predicted output history via LSTM hidden state $h(\mathbf{y}_{<t})$. Thus for a decoder with parameters $\theta$,

$$\log \phi_t(\mathbf{y}; \mathbf{x}; \theta) = f(\mathbf{x}; h(\mathbf{y}_{<t}); y_t; \theta).$$

Since, the local normalizer is easy to compute, likelihood maximization based training is a standard approach for training these models. This is also referred to as *cross-entropy* loss minimization or *teacher-forcing*. As mentioned earlier, this typical method for training locally normalized is not *search-aware* and suffers from *exposure bias*. This thesis explores two solutions to make the training search-aware while keeping the whole training procedure end-to-end (sub)-differentiable.

## 2.3 Globally Normalized Sequence models

In contrast, under a globally normalized model $\mathcal{M}_G$, the probability of $\mathbf{y}$ given $\mathbf{x}$ is computed via global scores  that condition on the whole sequence $\mathbf{y}$, instead of just the history:

$$p_{M_G}(\mathbf{y} \mid \mathbf{x}) = \frac{\prod_{t=1}^{T} {}_t(\mathbf{x}; \mathbf{y}_{nt}; y_t)}{Z_G(\mathbf{x})}$$

where $Z_G(\mathbf{x}) = \sum_{y \in Y} \prod_{t=1}^{T} s(\mathbf{x}; \mathbf{y}_{nt}; \mathbf{y}_t)$, is the global normalizer. $Z_G(\mathbf{x})$ is intractable to estimate for most problems of interest due to the large search space therefore, an exact likelihood maximization training approach is intractable for these models and we have to rely on inexact search. In our experiments, we show that globally normalized models are more amenable to recover from search errors during training in the context of inexact search and also ameliorate the issues related to *label bias* commonly encountered with locally normalized models. Also, these models naturally lend to computation of marginal probabilities of subsets of nodes in a sequence and hence are more amenable to train with external constraints on the posterior distribution. It must be noted that the globally normalized models in **Chapter 5** compute the scores ${}_t$ at each step by conditioning only on the tokens generated preceding the position $t$ i.e. ${}_t(\mathbf{x}; \mathbf{y}_{nt}; y_t) = {}_t(\mathbf{x}; \mathbf{y}_{<t}; y_t)$, they are still globally normalized by the virtue of the intractable normalizer $Z_G(\mathbf{x})$.

## 2.4 Energy-based Neural Models

The above-mentioned description of globally normalized models can also be interpreted as associating energy values with each possible sequence that are responsible for inducing a probability distribution over the space of all the possible sequences. The objective for training energy based models involves minimizing the energy of the observed data. For a finite set of sequences associated with positive scores, there is no distinction between the class of globally normalized models and the class of energy-based models. Let $p(\mathbf{y} \mid \mathbf{x};\ )$ be the probability of the sequence $y \in \mathcal{Y}$ conditioned

on the input $\mathbf{x}$ under the target distribution defined by the energy function $E(\mathbf{y}; \mathbf{x};\ )$ parametrized by , defined as follows:

$$p(\mathbf{y} \mid \mathbf{x};\ ) = \frac{e^{\ E(\mathbf{y};\mathbf{x};\ )}}{\sum_{\mathbf{y}' \in Y} e^{\ E(\mathbf{y}';\mathbf{x};\ )}} = \frac{(\mathbf{y}; \mathbf{x};\ )}{Z(\mathbf{x};\ )}$$

where represents the unnormalized score of the sequence $\mathbf{y}$ and $Z(\mathbf{x};\ )$ is the intractable normalization constant computed by summing over all the sequences.

In the rest of the document, no distinction is made between the terms *globally normalized models*, *unnormalized models*, and e*nergy-based models* and they are used interchangeably.

## 2.5 Expressivity: Autoregressive vs. Globally normalized models

While we have empirically discovered undesirable behavior exhibited by locally normalized models and look toward the class of globally normalized models as a possible solution, a natural question to ask is if the globally normalized models are inherently more expressive than the locally normalized models. More specifically, if $\mathcal{P}$ denotes the set of probability distributions over all the possible sequences upto a maximum length $L$, and the neural encoders and decoder have sufficiently high capacity, then whether the set of distributions $p_L$, induced by all the parametrizations of the locally normalized models equal to the set of distributions $p_G$, induced by all the parametrizations of the globally normalized models. For preneural sequence-to-sequence models, (Lafferty et al., 2001) showed that Maximum Entropy Markov Models (MEMMs) which is a class of locally normalized models, suffer from label bias and are subsets in terms of expressivity, of comparably featurized Conditional Random Fields (CRFs) which is a class of globally normalized models. This result has been influential in establishing the theoretical superiority of globally normalized models over the class of locally normalized models in prior work. However, this argument is not sufficient

and suitable for high-capacity neural sequence models. The major difference between preneural sequence models and contemporary neural sequence models is the expressive power over the representation of the conditioning input $\mathbf{x}$ for conditional generation. Pre-neural models typically had limited capacity to represent the input $\mathbf{x}$ while computing a local score for the output token $\mathbf{y}_t$ at each generation step $t$ i.e. in most cases it was not possible to have access to all of the input $\mathbf{x}$ while computing a score for fitness (Liang et al., 2008) of a particular output token during a generation step. The most common scenario was to limit the availability of $\mathbf{x}$ to just the prefix ($\mathbf{x}_{<t}$) for computing $\ _t(\mathbf{x}; y_{<t}; y)$ for both MEMMs and CRFs. This limitation is crucial to demonstrate label bias and expressive inferiority of the locally normalized models. However, contemporary neural encoders enable conditioning on powerful representations of the whole input $\mathbf{x}$ for score computation at all the token positions in the output $\mathbf{y}$.

Inspired by the theoretical results that establish the equivalence of probabilistic context free grammars and weighted context free grammars (Smith and Johnson, 2007; Chi, 1999; Abney et al., 1999), this document theoretically establishes that neural locally normalized models should be able to model any probability distribution that can be modelled by the neural globally normalized sequence models, and hence are equally expressive given sufficiently high capacity neural parametrization. I make certain reasonable assumptions to establish this equivalence:

- Finite sequence space: This is satisfied because in this document, we are concerned with sequences that reach a finite maximum length.
- Finite partition function: This assumption ensures that the globally normalized functions have a finite normalizer to induce a proper probability distribution over the countable sequences.

For each sequence in the space of sequences $\mathbf{y} \in \mathcal{Y}$, a globally normalized model associates a non-negative score $\ (\mathbf{y}; \ _g)$, which represents the probability of $\mathbf{y}$ as $\frac{(\mathbf{y}; \ _g)}{Z(\ _g)}$.[1] Since the number of sequences is countable and finite, we can define a function mapping $m : \mathbb{V}^n \rightarrow \mathbb{N}\ \forall n \in \mathbb{N};\ n \leq T$. Hence, this model can be represented as a function $g(\ _g) : \mathbb{N} \rightarrow [0; 1]$ such that $\sum_t g(t) \leq 1.0\ \forall\ t \in$

[1] We have suppressed the explicit dependence on the input sequence $\mathbf{x}$ here.

$\mathbb{N}$. Neurally parametrized locally normalized models naturally define a probability function on the space of sequences $p(\ _l) : \mathbb{N} \rightarrow [0; 1]$. Using the results of universal function approximation and Turing completeness of RNNs (Siegelmann and Sontag, 1995; Weiss et al., 2018) and autoregressive transformers (Pérez et al., 2019), it follows that $\exists\ _l$ such that $p(t) = g(t)\ \forall t \in \text{domain}(g)$. Therefore for any sequence $\mathbf{y}\ in \mathcal{Y}$, we can find a autoregressive parametrization $\ _l$ for a function $p$ such that $p(m(\mathbf{y})) = g(m(\mathbf{y}))$. In fact, because the partition function $Z_{\ g}$ is a free parameter in globally normalized models, there exist multiple parametrizations $\ _g$ that map to the same distribution over the sequences. This also means that an autoregressive model parametrized by $\ _l$ generally maps to multiple different globally normalized models parametrized by $\ _g$ which leads to issues pertaining to model identifiability. Therefore, as discussed in rest of the document, inductive bias associated with model classes plays a huge role in selecting appropriate models that, in addition to maximizing likelihood over the training data, also exhibit other desirable properties.

Despite this result of equivalence between autoregressive and globally normalized models, in **Chapter 5** we show that autoregressive models exhibit undesirable effects of label bias, although more subtly than their preneural counterparts, and hence are inferior to the comparable globally normalized models. This is due to the fact that the training and decoding objectives for neural models are always insufficiently optimized and results in learning miscalibrated distributions (Jiang et al., 2021) over sequences an tokens i.e. due to the presence of saddle points, multiple local optima, and even multiple global optima, it is difficult to exactly train neural models that not only have zero loss on the training set,but also exhibit all the desired phenomena. The models that are eventually used are heavily affected by the inductive bias that class of models being trained. Our empirical results show that globally normalized parametrization have better suited inductive biases that nudge the training of model toward parameters corresponding to well-behaved models. Similarly the decoding objective, typically maximum likelihood, is also almost never achieved exactly because of computational intractability of exact decoding procedures, which directly affects the quality of predictions that can be made from the models. As described in **Chapters 3–5**, this also implies the

need of *search-aware* training procedures which take into account the inexactness of the decoding procedures to be used during deployment. Therefore in the context of inexact training and decoding, the empirical results in this thesis demonstrate the superiority of globally normalized models over the autoregressive models despite them being equally expressive theoretically.

Another perspective for theoretical comparison of autoregressive models and globally normalized models is explored in Lin et al. (2021). Using a special construction, they show existence of distribution over sequences for which the unnormalized energy of the sequences can be computed in polynomial time, but the autoregressive conditional probabilities $p(y \mid \mathbf{y}_{<t})$ is NP-hard to compute wr.t the length of the sequence. Assuming $P \neq NP$, they use this argument to show that the size of the locally normalized models (in terms of parameters) must grow superpolynomially in the length of the sequences in order to compute the autoregressive factor and the probability of sequences in polynomial time. This is not the case for energy based models that can represent unnormalized distributinos over sequences with compact parametrization with polynomial time computation in sequence length. This theoretically exhibits a serious limitation of autoregressive models when it comes to very long or infinitely long sequences.

However, for the analysis in this thesis, this concern is downplayed by the fact that we restrict the maximum length of the sequences in order to be practically be able to model single sentences. Another factor that might alleviate this concern is the nature of true distribution over the natural language sequences for NLP applications considered in this thesis. For example, if the true underlying distribution is not as pathological as the construction in Lin et al. (2021), then it might be the case that it is possible for locally normalized models to reasonably approximate autoregressive conditionals in polynomial time with a compact parametrization. These considerations, coupled with powerful parametrization of the neural decoders enables us to focus on the limitations of autoregressive models that are different from the ones established in Lin et al. (2021). To conclude, the limitations explored in this thesis and Lin et al. (2021) indicate that the class of globally normalized models, in spite of being computationally expensive models, is a powerful class of

sequence models with many desirable properties over locally normalized autoregressive models.

# Chapter 3

# Ameliorating Exposure Bias: Differentiable Scheduled Sampling

In this chapter, we focus on training of search-aware locally normalized neural sequence models. We identify that a common cross-entropy optimization algorithm for training locally normalized sequence models suffers from *exposure bias* and propose objective functions that take the model's predictions into account for ameliorating this bias while maintaining end-to-end (sub)-differentiability of the objective so that training can be performed via automatic differentiation.

## 3.1 Standard Cross-entropy training

A common method to train neural sequence models is maximizing likelihood of target sequences $\mathbf{y}$ , conditioned on the input sequence $\mathbf{x}$ in a locally normalized model. The input is encoded via an expressive neural parametrization, typically a bidirectional LSTM to get an encoded representation of the input. A neural recurrent decoder then generates the output sequence in a left-to-right manner

with the log probability of the sequence computed using chain-rule as

$$\log p(\mathbf{y} \mid \mathbf{x}) = \sum_{i=1}^{|\mathbf{y}|} \log p(\mathbf{y}_i \mid \mathbf{y}_{1:i-1}; \mathbf{x}) \tag{3.1}$$

It is important to notice that in the formulation above, the probability of the target label at a particular decoding step is conditioned on the gold target history. This means that in recurrent decoder like an LSTM, the dynamics of hiddden states is guided by assuming access to the gold target history. Another possible parametrization for training could be to condition the probability of the gold target on each step on the input sequence and the model 's predicted history i.e.:

$$\log p(\mathbf{y} \mid \mathbf{x}) = \sum_{i=1}^{|\mathbf{y}|} \log p(\mathbf{y}_i \mid \hat{\mathbf{y}}_{1:i-1}; \mathbf{x}) \tag{3.2}$$

Equation 2.1 is the standard method to train the locally normalized model because it is naturally end-to-end differentiable and has shown to result in well-trained discriminative sequence models. Equation 2.2 reasons about model's behavior over its possible predictions in past while deciding about current prediction. We call the objective in Equation 2.2 *search-aware* because it is aware of inference with the model. As we discus in the following sections, search-aware training poses some challenges for training related to differentiability and tractability but it also has potential to result in more robust and well-informed models.

The objectives above, are similar to objective of well-known *Maximum Entropy Markov Models* (MEMMs)(McCallum et al., 2000), which focus on optimizing an independent classification loss at each decode-step conditioning the prediction on the input $\mathbf{x}$ and the history $\mathbf{y}_{<}$. The neural LSTM parametrization allows us to condition on *full* input and *complete* history, which makes these models very expressive.

## 3.2 Exposure Bias

The standard cross-entropy training procedure described above always uses gold contexts instead of model's predicted context during training of the recurrent parameters of the decoder. Hence, the states and contexts encountered during training do not match those encountered at decode time and it is likely for the decoder to encounter recurrent hidden states that it never encountered during training making it difficult to recover from errors in model's predictions during decoding. This phenomenon of mismatch between training and decoding phases is referred to as *exposure bias*(Ranzato et al., 2016). It has been observed that the models that use a more advanced inference procedure than greedy decoding like beam search during decoding, but do not account for it during training sometimes yield reduced test performance when compared with greedy decoding (Koehn and Knowles, 2017; Neubig, 2017; Cho et al., 2014). This issue has been addressed using several approaches that try to incorporate awareness of decoding choices into the training optimization. These include reinforcement learning (Ranzato et al., 2016; Bahdanau et al., 2017), imitation learning (Daumé et al., 2009; Ross et al., 2011; Bengio et al., 2015), beam-search based approaches (Wiseman and Rush, 2016; Andor et al., 2016; Daumé III and Marcu, 2005) and other approaches that focus on decoding procedures based on variational approximation (Gormley et al., 2015). *scheduled sampling* (Bengio et al., 2015) is another simple approach designed to counter exposure bias which stochastically incorporates contexts from previous decoding decisions into training.

In this chapter, we propose a solution to ameliorate exposure bias that is inspired from a technique called 'scheduled sampling' which involves mixing in models predictions instead of the gold labels at random decoding steps to be fed in as the input embedding at the next step. This approach while shown to work has a major flaw in that it makes the objective discontinuous wrt. the model parameters and hence not amenable for end-to-end training via an automatic differentiation toolkit. We present an algorithm that is end-to-end differentiable and is inspired by scheduled sampling. Empirical comparison on Machine Translation (MT) and tagging tasks like Named

entity recognition exhibits the empirical superiority of our search-aware solution over standard cross-entropy based training baselines.

## 3.3 Differentiable Scheduled Sampling

One the simplest to implement and least computationally expensive approaches to ameliorate exposure bias is *scheduled sampling* (Bengio et al., 2015), which stochastically incorporates contexts from previous decoding decisions into training. While scheduled sampling has been empirically successful, its training objective has a drawback: because the procedure directly incorporates greedy decisions at each time step, the objective is discontinuous at parameter settings where previous decisions change their value. As a result, gradients near these points are non-informative and scheduled sampling has difficulty assigning credit for errors. In particular, the gradient does not provide information useful in distinguishing between local errors without future consequences and cascading errors which are more serious.

Hence in this thesis, a novel approach based on scheduled sampling is proposed that uses a differentiable approximation of previous greedy decoding decisions inside the training objective by incorporating a continuous relaxation of argmax. As a result, our end-to-end relaxed greedy training objective is differentiable everywhere and fully continuous. By making the objective continuous at points where previous decisions change value, this approach provides gradients that can respond to cascading errors. In addition, we demonstrate a related approximation and reparametrization for sample-based training (another training scenario considered by scheduled sampling (Bengio et al., 2015)) that can yield stochastic gradients with lower variance than in standard scheduled sampling. In the experiments on two different tasks, machine translation (MT) and named entity recognition (NER), we show that our approach outperforms both cross-entropy training and standard scheduled sampling procedures with greedy and sampled-based training.

### 3.3.1 Discontinuity in Scheduled Sampling

While scheduled sampling (Bengio et al., 2015) is an effective way to rectify exposure bias, it cannot differentiate between cascading errors, which can lead to a sequence of bad decisions, and local errors, which have more benign effects. Specifically, scheduled sampling focuses on learning optimal behavior in the current step given the fixed decoding decision of the previous step. If a previous bad decision is largely responsible for the current error, the training procedure has difficulty adjusting the parameters accordingly. The following machine translation example highlights this credit assignment issue:

Ref: The cat purrs . Pred: The dog barks .

At step 3, the model prefers the word 'barks' after incorrectly predicting 'dog' at step 2. To correct this error, the scheduled sampling procedure would increase the score of 'purrs' at step 3, conditioned on the fact that the model predicted (incorrectly) 'dog' at step 2, which is not the ideal learning behaviour. Ideally, the model should be able to backpropagate the error from step 3 to the source of the problem which occurred at step 2, where 'dog' was predicted instead of 'cat'. The lack of credit assignment during training is a result of discontinuity in the objective function used by scheduled sampling, as illustrated in Figure 3.1. We denote the ground truth target symbol at step $i$ by $y_i$, the embedding representation of word $y$ by $e(y)$, and the hidden state of a seq2seq decoder at step $i$ as $h_i$. Standard cross-entropy training defines the loss at each step to be $\log p(y_i|h_i(e(y_{i-1}); h_{i-1}))$, while scheduled sampling uses loss $\log p(y_i|h_i(e(\hat{y}_{i-1}); h_{i-1}))$, where $\hat{y}_{i-1}$ refers the model's prediction at the previous step.[1] Here, the model prediction $\hat{y}_{i-1}$ is obtained by performing an argmax over the output softmax layer.

Hence, in addition to the intermediate hidden states and final softmax scores, the previous model prediction, $\hat{y}_{i-1}$, itself depends on the model parameters, , and ideally, should be backpropagated through, unlike the gold target symbol $y_{i-1}$ which is independent of model parameters. However,

[1]For the sake of simplicity, the 'always sample' variant of scheduled sampling is described (Bengio et al., 2015).

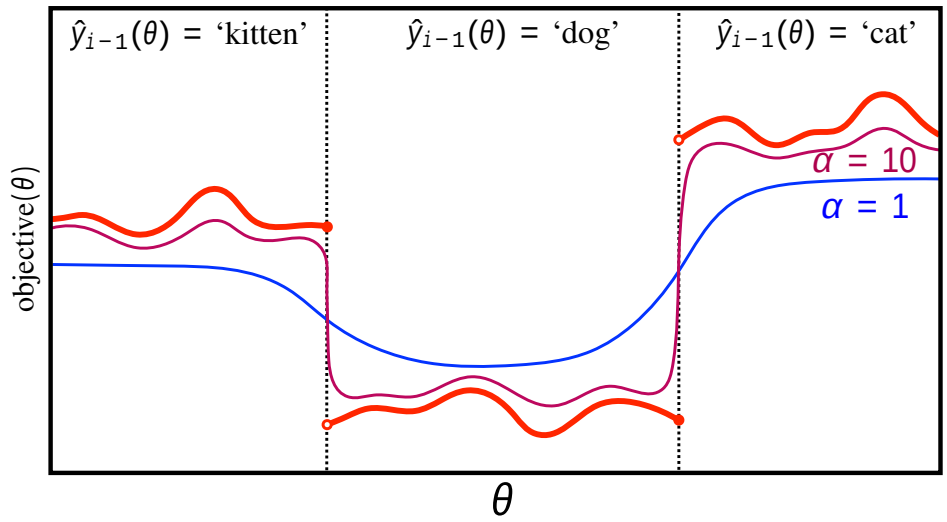


**Figure 3.1:** Discontinuous scheduled sampling objective (red) and continuous relaxations (blue and purple).

the argmax operation is discontinuous, and thus the training objective (depicted in Figure 3.1 as the red line) exhibits discontinuities at parameter settings where the previous decoding decisions change value (depicted as changes from 'kitten' to 'dog' to 'cat'). Because these change points represent discontinuities, their gradients are undefined and the effect of correcting an earlier mistake (for example 'dog' to 'cat') as the training procedure approaches such a point is essentially hidden. In our approach, described in detail in the next section, we attempt to fix this problem by incorporating a continuous relaxation of the argmax operation into the scheduled sampling procedure in order to form an approximate but fully continuous objective. Our relaxed approximate objective is depicted in Figure 3.1 as blue and purple lines, depending on temperature parameter which trades-off smoothness and quality of approximation.

### 3.3.2 Credit Assignment via Relaxation

In this section we explain in detail the continuous relaxation of greedy decoding that we will use to build a fully continuous training objective. We also introduce a related approach for sample-based training.

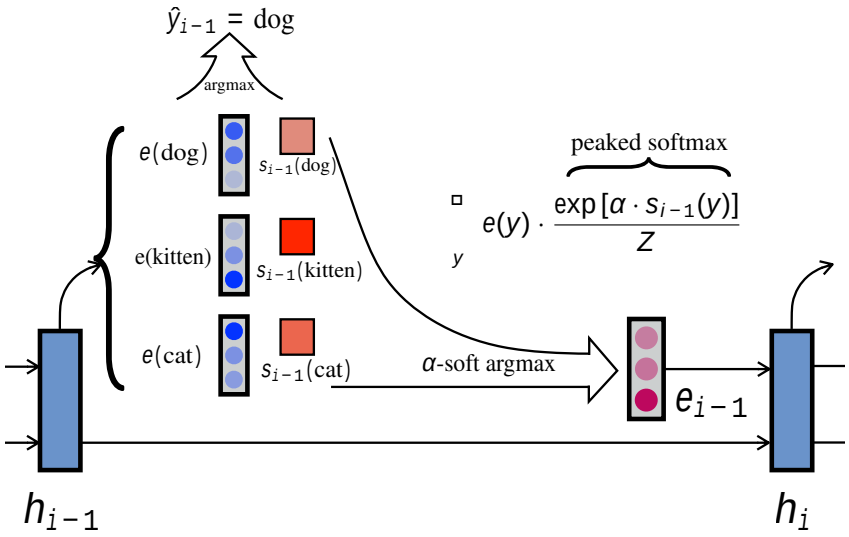


**Figure 3.2:** Relaxed greedy decoder that uses a continuous approximation of argmax as input to the decoder state at next time step.

### Soft Argmax

In scheduled sampling, the embedding for the best scoring word at the previous step is passed as an input to the current step. This operation[2] can be expressed as

$$\hat{e}_{i-1} = \sum_y e(y) \mathbf{1}[\forall y' \neq y \; s_{i-1}(y) > s_{i-1}(y')]$$

where $y$ is a word in the vocabulary, $s_{i-1}(y)$ is the output score of that word at the previous step, and $\hat{e}_{i-1}$ is the embedding passed to the next step. This operation can be relaxed by replacing the indicator function with a *peaked softmax function* with hyperparameter to define a soft argmax procedure:

$$e_{i-1} = \sum_y e(y) \cdot \frac{\exp( s_{i-1}(y))}{\sum_{y'} \exp( s_{i-1}(y'))}$$

As $\to \infty$, the equation above approaches the true argmax embedding. Hence, with a finite and large , we get a linear combination of all the words (and therefore a continuous function of the

[2]Assuming there are no ties for the sake of simplicity.

parameters) that is dominated heavily by the word with maximum score.

### Soft Reparametrized Sampling

Another variant of scheduled sampling is to pass a sampled embedding from the softmax distribution at the previous step to the current step instead of the argmax. This is expected to enable better exploration of the search space during optimization due to the added randomness and hence result in a more robust model. In this section, we discuss and review an approximation to the Gumbel reparametrization trick that we use as a module in our sample-based decoder. This approximation was proposed by Maddison et al. (2017) and Jang et al. (2016), who showed that the same soft argmax operation introduced above can be used for reducing variance of stochastic gradients when sampling from softmax distributions. Unlike soft argmax, this approach is not a fully continuous approximation to the sampling operation, but it does result in much more informative gradients compared to naive scheduled sampling procedure.

The Gumbel reparametrization trick shows that sampling from a categorical distribution can be refactored into sampling from a simple distribution followed by a deterministic transformation as follows: (i) sampling an independent Gumbel noise $G$ for each element in the categorical distribution, typically done by transforming a sample from the uniform distribution: $U \sim Uniform(0; 1)$ as $z$, then (ii) adding it componentwise to the unnormalized score of each element, and finally (iii) taking an argmax over the vector. Using the same argmax softening procedure as above, they arrive at an approximation to the reparametrization trick which mitigates some of the gradient's variance introduced by sampling. The approximation is[3]:

$$e_{i\ 1} = \sum_{y} e(y) \cdot \frac{\exp(\quad (s_{i\ 1}(y) + G_y))}{\sum_{y'} \exp(\quad (s_{i\ 1}(y') + G_{y'}))}$$

We will use this 'concrete' approximation of softmax sampling in our relaxation of scheduled

[3]This is different from using the expected softmax embedding because our approach approximates the actual sampling process instead of linearly weighting the embeddings by their softmax probabilities

sampling with a sample-based decoder. We discuss details in the next section. Note that our original motivation based on removing discontinuity does not strictly apply to this sampling procedure, which still yields a stochastic gradient due to sampling from the Gumbel distribution. However, this approach is conceptually related to greedy relaxations since, here, the soft argmax reparametrization reduces gradient variance which may yield a more informative training signal. Intuitively, this approach results in the gradient of the loss to be more aware of the sampling procedure compared to naive scheduled sampling and hence carries forward information about decisions made at previous steps. The empirical results, discussed later, show similar gains to the greedy scenario.

**Differentiable Relaxed Decoders**

With the argmax relaxation introduced above, we have a recipe for a fully differentiable greedy decoder designed to produce informative gradients near change points. Our final training network for scheduled sampling with relaxed greedy decoding is shown in Figure 3.2. Instead of conditioning the current hidden state, $h_i$, on the argmax embedding from the previous step, $\hat{e}_{i-1}$, we use the $\tau$-soft argmax embedding, $e_{i-1}$, defined in Section 3.3.2. This removes the discontinuity in the original greedy scheduled sampling objective by passing a linear combination of embeddings, dominated by the argmax, to the next step. Figure 3.1 illustrates the effect of varying $\tau$. As $\tau$ increases, we more closely approximate the greedy decoder.

As in standard scheduled sampling, here we minimize the cross-entropy based loss at each time step. Hence the computational complexity of our approach is comparable to standard seq2seq training. As we discuss in Section 3.3.3, mixing model predictions randomly with ground truth symbols during training (Bengio et al., 2015; Daumé et al., 2009; Ross et al., 2011), while annealing the probability of using the ground truth with each epoch, results in better models and more stable training. As a result, training is reliant on the *annealing schedule* of two important hyperparameters: i) ground truth mixing probability and ii) the $\tau$ *parameter* used for approximating the argmax function. For output prediction, at each time step, we can still output the hard argmax, depicted in

Figure 3.2.

For the case of scheduled sampling with sample-based training–where decisions are sampled rather than chosen greedily (Bengio et al., 2015)–we conduct experiments using a related training procedure. Instead of using soft argmax, we use the soft sample embedding, $\mathbf{e}_{i\ 1}$, defined in Section 3.3.2. Apart from this difference, training is carried out using the same procedure.

### 3.3.3 Experimental Setup

We perform experiments with machine translation (MT) and named entity recognition (NER).

#### Data

For MT, we use the same dataset (the German-English portion of the IWSLT 2014 machine translation evaluation campaign (Cettolo et al., 2014)), preprocessing and data splits as Ranzato et al. (2016). For named entity recognition, we use the CONLL 2003 shared task data (Tjong Kim Sang and De Meulder, 2003) for German language and use the provided data splits. We perform no preprocessing on the data.The output vocabulary length for MT is 32000 and 10 for NER.

#### Implementation details

For MT, we use a seq2seq model with a simple attention mechanism (Bahdanau et al., 2015), a bidirectional LSTM encoder (1 layer, 256 units), and an LSTM decoder (1 layer, 256 units). For NER, we use a seq2seq model with an LSTM encoder (1 layer, 64 units) and an LSTM decoder (1 layer, 64 units) with a fixed attention mechanism that deterministically attends to the $i$th input token when decoding the $i$th output, and hence does not involve learning of attention parameters. [4]

[4] *Fixed attention* refers to the scenario when we use the bidirectional LSTM encoder representation of the source sequence token at time step $t$ while decoding at time step $t$ instead of using a linear combination of all the input

**Hyperparameter tuning**

We start by training with actual ground truth sequences for the first epoch and decay the probability of selecting the ground truth token as an inverse sigmoid (Bengio et al., 2015) of epochs with a decay strength parameter $k$. We also tuned for different values of and explore the effect of varying exponentially (annealing) with the epochs. In table 3.1, we report results for the best performing configuration of decay parameter and the parameter on the validation set. To account for variance across randomly started runs, we ran multiple random restarts (RR) for all the systems evaluated and always used the RR with the best validation set score to calculate test performance.

**Comparison**

We report validation and test metrics for NER and MT tasks in Table 3.1, F1 and BLEU respectively. 'Greedy' in the table refers to scheduled sampling with soft argmax decisions (either soft or hard) and 'Sample' refers the corresponding reparametrized sample-based decoding scenario. We compare our approach with two baselines: standard cross-entropy loss minimization for seq2seq models ('Baseline CE') and the standard scheduled sampling procedure (Bengio et al. (2015)). We report results for two variants of our approach: one with a fixed parameter throughout the training procedure ( -soft fixed), and the other in which we vary exponentially with the number of epochs ( -soft annealed).

| Training procedure | NER (F1) | | | | MT (BLEU) | | | |
|---|---|---|---|---|---|---|---|---|
| | Dev | | Test | | Dev | | Test | |
| Baseline CE | 49.43 | | 53.32 | | 20.35 | | 19.11 | |
| | Greedy | | Sample | | Greedy | | Sample | |
| | Dev | Test | Dev | Test | Dev | Test | Dev | Test |
| Bengio et al. (2015) | 49.75 | 54.83 | 50.90 | 54.60 | 20.52 | 19.85 | 20.40 | 19.69 |
| -soft fixed | 51.65 | 55.88 | 51.13 | 56.25 | 21.32 | 20.28 | 20.48 | 19.69 |
| -soft annealed | 51.43 | **56.33** | 50.99 | 54.20 | 21.28 | 20.18 | 21.36 | **20.60** |

**Table 3.1:** Result on NER and MT. We compare our approach ( -soft argmax with fixed and annealed temperature) with standard cross entropy training (Baseline CE) and discontinuous scheduled sampling (Bengio et al. (2015)). 'Greedy' and 'Sample' refer to Section 3.3.2 and Section 3.3.2.

### 3.3.4 Results

All three approaches improve over the standard cross-entropy based seq2seq training. Moreover, both approaches using continuous relaxations (greedy and sample-based) outperform standard scheduled sampling (Bengio et al., 2015). The best results for NER were obtained with the relaxed greedy decoder with annealed which yielded an F1 gain of +3.1 over the standard seq2seq baseline and a gain of +1.5 F1 over standard scheduled sampling. For MT, we obtain the best results with the relaxed sample-based decoder, which yielded a gain of +1.5 BLEU over standard seq2seq and a gain of +0.75 BLEU over standard scheduled sampling.

We observe that the reparametrized sample-based method, although not fully continuous end-to-end unlike the soft greedy approach, results in good performance on both the tasks, particularly MT. This might be an effect of stochastic exploration of the search space over the output sequences during training and hence we expect MT to benefit from sampling due to a much larger search space associated with it. We also observe that annealing results in good performance which suggests that a smoother approximation to the loss function in the initial stages of training is helpful in guiding the learning in the right direction. However, in our experiments we noticed that the performance while annealing was sensitive to the hyperparameter associated with the annealing

sequences weighted according to the attention parameters in the standard attention mechanism based models.

| k | 100 | 10 | 1 | *Always* |
|---|---|---|---|---|
| NER (F1) | 56.33 | 55.88 | 55.30 | 54.83 |

**Table 3.2:** Effect of different schedules for scheduled sampling on NER. k is the decay strength parameter. Higher k corresponds to gentler decay schedules. *Always* refers to the case when predictions at the previous predictions are always passed on as inputs to the next step.

schedule of the mixing probability in scheduled sampling during training.

The computational complexity of our approach is comparable to that of standard seq2seq training. However, instead of a vocabulary-sized max and lookup, our approach requires a matrix multiplication. Practically, we observed that on GPU hardware, all the models for both the tasks had similar speeds which suggests that our approach leads to accuracy gains without compromising run-time. Moreover, as shown in Table 3.2, we observe that a gradual decay of mixing probability consistently compared favorably to more aggressive decay schedules. We also observed that the 'always sample' case of relaxed greedy decoding, in which we never mix in ground truth inputs (see Bengio et al. (2015)), worked well for NER but resulted in unstable training for MT. We reckon that this is an effect of large difference between the search space associated with NER and MT.

### 3.3.5 Conclusion

Our positive results indicate that mechanisms for credit assignment can be useful when added to the models that aim to ameliorate exposure bias. Further, our results suggest that continuous relaxations of the argmax operation can be used as effective approximations to hard decoding during training.

# Chapter 4

# Ameliorating Exposure Bias: Continuous Relaxation to Beam Search

This chapter introduces another solution to ameliorate exposure bias that is specifically applicable to the setting in which beam search is the decoding algorithm in use during deployment of a seqeunce model. It involves making the model aware to beam search while training. Because the outputs of the sequence models are discrete and the beam search procedure, commonly used during inference, is itself discontinuous, there is no straightforward way to incorporate beam search into training in an end-to-end (sub)-differentiable manner. We propose a continuous relaxation to the beam search procedure and such that we can interpret the beam-search aware objective as a computation graph and train the model using automatic differentiation libraries. Experiments on natural language processing tasks like Named Entity Recognition, and CCG supertagging demonstrate the effectiveness of this approach over standard autoregressive training baselines.

## 4.1 Beam Search-aware Training

In the previous section on differentiable scheduled sampling, techniques to ameliorate exposure bias when the inference during decoding was either greedy step-wise or involved sampling tokens at each step during decoding. However, *Beam search* is a desirable choice of test-time decoding algorithm for neural sequence models because it potentially avoids search errors made by simpler greedy methods. When *beam search* is used for decoding, the problems associated with exposure bias are likely to amplify because the teacher-forcing based training is performed in a manner that is drastically different from beam search. As a result, for cross-entropy trained models with teacher forcing, beam decoding has been observed to yield reduced test performance when compared with greedy decoding (Koehn and Knowles, 2017; Neubig, 2017; Cho et al., 2014). Therefore, in order to train models that can more effectively make use of beam search, we introduce, in this section, a new training procedure that focuses on the final loss metric (e.g. Hamming loss) evaluated on the output of beam search. While well-defined and a valid training criterion, this "direct loss" objective is discontinuous and thus difficult to optimize. Hence, in our approach, we form a sub-differentiable surrogate objective by introducing a novel continuous approximation of the beam search decoding procedure.

## 4.2 Beam Search

The beam search procedure for obtaining the k-best sequences from a recurrent neural decoder is described in Algorithm 1. The variables are subscripted by decoding steps and the beam element. $h$ refers to the hidden state of the recurrent decoder, $f$ refers to the local score producing function involving recurrent network at each step. Each beam element is associated with a recurrent decoder. Therefore, a beam of size $k$ will have $k$ recurrent decoders. At each step, each element in the beam yields scores for candidate successors. The top k-scoring successors are chosen and their parent beam elements are stored as backpointers. The chosen successors further the recurrent network

dynamics at each beam element. Once the search is run over maximum steps, the backpointers are followed from step $T$ to step 1 to yield a sequence.

**Algorithm 1** Standard Beam Search

**Initialize:**
  $h_{0;i} \leftarrow \vec{0}$, $e_{0;i} \leftarrow embedding([START])$, $s_{0;i} \leftarrow 0$, $i = 1;:::;k$
**for** t = 0 to T **do**
  **for** i = 1 to k **do**
    **for all** $v \in V$ **do**
      $s_t[i;v] \leftarrow s_{t;i} + f(h_{t;i};v)$ . $f$ is the local output scoring function
  $s_{t+1} \leftarrow top\text{-}k\text{-}max(s_t)$ . Top $k$ values of the input matrix
  $b_{t+1;} ; y_{t;} \leftarrow top\text{-}k\text{-}argmax(s_t)$ . Top $k$ argmax index pairs of the input matrix
  **for** i = 1 to k **do**
    $e_{t+1;i} \leftarrow embedding(y_{t;i})$
    $h_{t+1;i} \leftarrow r(h_{t;i};e_{t+1;i})$ . $r$ is a nonlinear recurrent function that returns state at next step
$\hat{y} \leftarrow follow\text{-}backpointer((b_{1;} ; y_{1;}); :::; (b_{T;} ; y_{T;}))$
$s(\hat{y}) \leftarrow max(s_T)$

## 4.3 Objective

We denote the seq2seq model parameterized by as $\mathcal{M}(\ )$. We denote the result of beam search over $\mathcal{M}(\ )$ as $\hat{\mathbf{y}} = Beam(\mathbf{x}; \mathcal{M}(\ ))$. Ideally, we would like to directly minimize a final evaluation loss, $L(\hat{\mathbf{y}}; \mathbf{y}\ )$, evaluated on the result of running beam search with input $x$ and model $\mathcal{M}(\ )$. For tractability, we assume that the evaluation loss decomposes over time steps $t$ as: $L(\hat{\mathbf{y}}; \mathbf{y}\ ) = \sum_{t=1}^{T} d(\hat{\mathbf{y}}_t; \mathbf{y}\ )$[1]. We refer to this idealized training objective that directly evaluates prediction loss as the "direct loss" objective and define it as:

$$\min G_{\mathrm{DL}}(\mathbf{x};\ ;\mathbf{y}\ ) = \min L(Beam(\mathbf{x}; \mathcal{M}(\ ));\mathbf{y}\ ) \tag{4.1}$$

[1]This assumption does not hold for some popular evaluation metrics (e.g. BLEU). In these cases, surrogate evaluation losses such as Hamming distance can be used .

Unfortunately, optimizing this objective using gradient methods is difficult because the objective is discontinuous. The two sources of discontinuity are:

1. Beam search decoding (referred to as the function *Beam*) involves discrete argmax decisions and thus represents a discontinuous function.
2. The output, $\hat{\mathbf{y}}$, of the *Beam* function, which is the input to the loss function, $L(\hat{\mathbf{y}}; \mathbf{y})$, is discrete and hence the evaluation of the final loss is also discontinuous.

We introduce a surrogate training objective that avoids these problems and as a result is fully continuous. In order to accomplish this, we propose a continuous relaxation to the *composition* of our final loss metric, $L$, and our decoder function, $Beam$:

$$softLB(\mathbf{x}; \mathcal{M}(\ ); \mathbf{y}) \approx (L \circ Beam)(\mathbf{x}; \mathcal{M}(\ ); \mathbf{y})$$

Specifically, we form a continuous function *softLB* that seeks to approximate the result of running our decoder on input $\mathbf{x}$ and then evaluating the result against $\mathbf{y}$ using $L$. By introducing this new module, we are now able to construct our surrogate training objective:

$$\min G_{\mathrm{DL}}(\mathbf{x}; \ ; \mathbf{y}) = \min softLB(\mathbf{x}; \mathcal{M}(\ ); \mathbf{y}) \tag{4.2}$$

Specified in more detail in Section 4.3.3, our surrogate objective in Equation 4.2 will additionally take a hyperparameter that trades approximation quality for smoothness of the objective. We first describe the standard discontinuous beam search procedure and then our training approach involving a continuous relaxation of beam search.

### 4.3.1 Discontinuity in Beam Search

Formally, *beam search* is a procedure with hyperparameter $k$ that maintains a beam of $k$ elements at each time step and expands each of the $k$ elements to find the $k$-best candidates for the next time

**Algorithm 2** continuous-top-k-argmax

**Inputs:**
$s \in \mathbb{R}^{|V|}$
**Outputs:**
$p_i \in \mathbb{R}^{|V|}$, s.t. $\sum_j p_{ij} = 1; i = 1; \ldots; k$
$m \in \mathbb{R}^k = \textit{top-k-max}(s)$
**for** $i = 1$ to k **do** . *peaked-softmax* will be dominated by scores closer to $m_i$
    $p_i = \textit{peaked-softmax}\ (-(s - m_i \cdot \mathbf{1})^2)$ . The square operation is element-wise

step. The procedure finds an approximate argmax of a scoring function defined on output sequences.

We describe beam search in the context of seq2seq models in Algorithm 1 – more specifically, for an encoder-decoder (Sutskever et al., 2014) model with a nonlinear auto-regressive decoder (e.g. an LSTM (Hochreiter and Schmidhuber, 1997)). We define the global model score of a sequence $y$ with length $T$ to be the sum of local output scores at each time step of the seq2seq model: $s(y) = \sum_{t=1}^{T} f(h_t; y_t)$. In neural models, the function $f$ is implemented as a differentiable mapping, $\mathbb{R}^{|h|} \rightarrow \mathbb{R}^{|V|}$, which yields scores for vocabulary elements using the recurrent hidden states at corresponding time steps. In our notation, $h_{t;i}$ is the hidden state of the decoder at time step $t$ for beam element $i$, $e_{t;i}$ is the embedding of the output symbol at time-step $t$ for beam element $i$, and $s_{t;i}$ is the cumulative model score at step $t$ for beam element $i$. In Algorithm 1, we denote by $\mathbf{s}_t \in \mathbb{R}^{k \times |V|}$ the cumulative candidate score matrix which represents the model score of each successor candidate in the vocabulary for each beam element. This score is obtained by adding the local output score (computed as $f(h_{t;i}; w)$) to the running total of the score for the candidate. The function $r$ in Algorithms 1 and 3 yields successive hidden states in recurrent neural models like RNNs, LSTMs etc. The *embedding* operation maps a word in the vocabulary $V$, to a continuous embedding vector. Finally, backpointers at each time step to the beam elements at the previous time step are also stored for identifying the best sequence $\hat{y}$, at the conclusion of the search procedure. A backpointer at time step $t$ for a beam element $i$ is denoted by $b_{t;i} \in \{1; \ldots; k\}$ which points to one of the $k$ elements at the previous beam. We denote a vector of backpointers for all the beam elements by $b_{t;}$ . The *follow-backpointer* operation takes as input backpointers ($b_{t;}$ ) and candidates

**Algorithm 3** Continuous relaxation to beam search

**Initialize:**
$h_{0;i} \leftarrow \vec{0}$, $e_{0;i} \leftarrow embedding([START])$, $s_{0;i} \leftarrow 0$, $D_t \in \mathbb{R}^k \leftarrow \vec{0}$,
$i = 1;:::;k$
**for** t = 0 to T **do**
  **for all** $w \in V$ **do**
    **for** i=1 to k **do**
      $s_t[i;w] \leftarrow s_{t;i} + f(h_{t;i};w)$ . $f$ is a local output scoring function
    $D_{t;w} = d(w)$ . $D_t$ is used to compute $D_{t+1}$
  $p_1;:::;p_k \leftarrow continuous\text{-}top\text{-}k\text{-}argmax(s_t)$ . Call Algorithm 2
  **for** i = 1 to k **do**
    $\vec{b}_{t;i} \leftarrow row\_sum(p_i)$ . Soft back pointer computation
    $a_i \in \mathcal{R}^{jVj} \leftarrow column\_sum(p_i)$ . Contribution from vocabulary items
    $e_{t+1;i} \leftarrow a_i^T \times E$ . Peaked distribution over the candidates to compute $e; D; S$
    $D_{t+1;i} \leftarrow a_i^T \cdot D_t$
    $s_{t+1;i} = sum(s_t \odot p_i)$
    $\vec{h}_{t;i} \leftarrow \vec{0}$
    **for** j = 1 to k **do** . Get contributions from soft backpointers for each beam element
      $\vec{h}_{t;i} += h_{t;j} * \vec{b}_{t;i}[j]$
      $D_{t+1;i} += D_{t;j} * \vec{b}_{t;i}[j]$
    $h_{t+1;i} \leftarrow r(\vec{h}_{t;i}; e_{t+1;i})$ . $r$ is a nonlinear recurrent function that returns state at next step
$L = peaked\text{-}softmax\ (s_T) \cdot D_T$ . Pick the loss for the sequence with highest model score on the beam in a soft manner.

($y_{t;} \in \{1;:::;|V|\}^k$) for all the beam elements at each time step and traverses the sequence in reverse (from time-step $T$ through 1) following backpointers at each time step and identifying candidate words associated with each backpointer that results in a sequence $\hat{y}$, of length $T$.

The procedure described in Algorithm 1 is discontinuous because of the *top-k-argmax* procedure that returns a pair of vectors corresponding to the $k$ highest-scoring indices for backpointers and vocabulary items from the score matrix $s_t$. This index selection results in hard backpointers at each time step which restrict the gradient flow during backpropagation. In the next section, we describe a continuous relaxation to the *top-k-argmax* procedure which forms the crux of our approach.

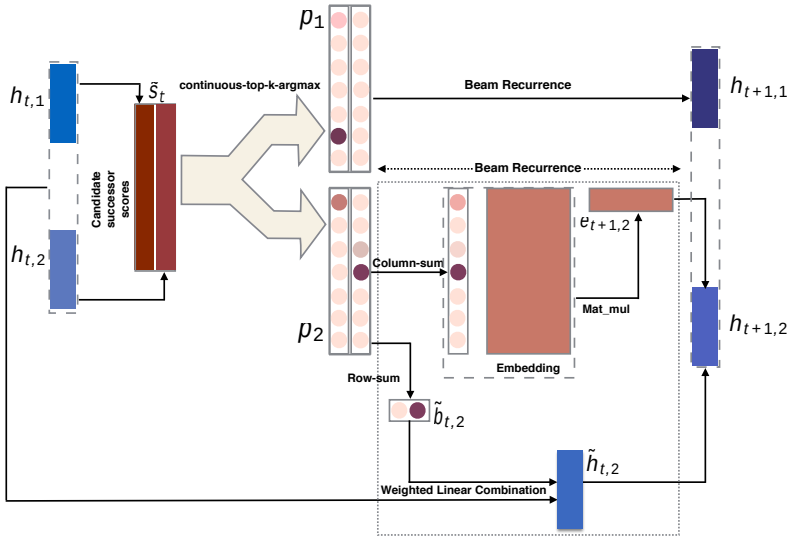


**Figure 4.1:** Illustration of our approximate continuous beam search (Algorithm 3) module to obtain hidden states for beam elements at the next time step ($h_{t+1;}$ ), starting from the hidden states corresponding to beam elements are current time step ($h_{t;}$ ) with beam size of 2. 'Beam recurrence' module has been expanded for $h_{t+1;2}$ and similar procedure is carried out for $h_{t+1;1}$.

### 4.3.2 Continuous Approximation to *top-k-argmax*

The key property that we use in our approximation is that for a real valued vector $z$, the argmax with respect to a vector of scores, s, can be approximated by a temperature controlled softmax operation. The argmax operation can be represented as:

$$\hat{z} = \sum_i z_i \mathbb{1}[\forall i' \neq i;\ s_i > s_{i'}];$$

which can be relaxed by replacing the indicator function with a *peaked-softmax* operation with hyperparameter :

$$\tilde{z} = \sum_i z_i \frac{\exp(\ s_i)}{\sum_{i'} \exp(\ s_{i'})} = z^T \cdot \frac{\text{elem-exp}(\ s)}{\sum_{i'} \exp(\ s_{i'})}$$

$$= z^T \cdot \textit{peaked-softmax}\ (\text{s})$$

As $\to \infty$, $\tilde{z} \to \hat{z}$ so long as there is only one maximum value in the vector $z$. This *peaked-softmax* operation has been shown to be effective in recent work (Maddison et al., 2017; Jang et al., 2016; Goyal et al., 2017) involving continuous relaxation to the argmax operation, although to our knowledge, this is the first work to apply it to approximate the beam search procedure.

Using this *peaked-softmax* operation, we propose an iterative algorithm for computing a continuous relaxation to the *top-k-argmax* procedure in Algorithm 2 which takes as input a score matrix of size $k \times |V|$ and returns $k$ peaked matrices $p$ of size $k \times |V|$. Each matrix $p_i$ represents the index of $i$-th max. For example, $p_1$ will have most of its mass concentrated on the index in the matrix that corresponds to the argmax, while $p_2$ will have most of its mass concentrated on the index of the 2nd-highest scoring element. Specifically, we obtain matrix $p_i$ by computing the squared difference between the $i$-highest score and all the scores in the matrix and then using the *peaked-softmax* operation over the negative squared differences. This results in scores closer to the $i$-highest score to have a higher mass than scores far away from the $i$-highest score.

Hence, the continuous relaxation to *top-k-argmax* operation can be simply implemented by iteratively using the *max* operation which is continuous and allows for gradient flow during back-propagation. As $\to \infty$, each $p$ vector converges to hard index pairs representing hard backpointers and successor candidates described in Algorithm 1. For finite , we introduce a notion of a soft backpointer, represented as a vector $\tilde{b} \in \mathbb{R}^k$ in the $k$-probability simplex, which represents the contribution of each beam element from the previous time step to a beam element at current time step. This is obtained by a row-wise sum over $p$ to get $k$ values representing soft backpointers.

### 4.3.3 Training with Continuous Relaxation of Beam Search

We describe our approach in detail in Algorithm 3 and illustrate the soft beam recurrence step in Figure 4.1. For composing the loss function and the beam search function for our optimization as proposed in Equation 2, we make use of decomposability of the loss function across time-steps. Thus for a sequence y, the total loss is: $L(\mathrm{y}; \mathrm{y}) = \sum_{t=1}^{T} d(y_t)$. In our experiments, $d(y_t)$ is the

Hamming loss which can be easily computed at each time-step by simply comparing gold $y_t$ with $y_t$. While exact computation of $d(y)$ will vary according to the loss, our proposed procedure will be applicable as long as the total loss is decomposable across time-steps. While decomposability of loss is a strong assumption, existing literature on structured prediction (Taskar et al., 2004; Tsochantaridis et al., 2005) has made due with this assumption, often using decomposable losses as surrogates for non-decomposable ones. We detail the continuous relaxation to beam search in Algorithm 3 with $D_{t;i}$ being the cumulative loss of beam element $i$ at time step $t$ and $E$ being the embedding matrix of the target vocabulary which is of size $|V| \times l$ where $l$ is the size of the embedding vector.

In Algorithm 3, all the discrete selection functions have been replaced by their soft, continuous counterparts which can be backpropagated through. This results in all the operations being matrix and vector operations which is ideal for a GPU implementation. An important aspect of this algorithm is that we no longer rely on exactly identifying a discrete search prediction $\hat{y}$ since we are only interested in a continuous approximation to the direct loss $L$ (line 18 of Algorithm 3), and all the computation is expressed via the *soft beam search* formulation which eliminates all the sources of discontinuities associated with the training objective in Equation 1. The computational complexity of our approach for training scales linearly with the beam size and hence is roughly $k$ times slower than standard CE training for beam size $k$. Since we have established the pointwise convergence of *peaked-softmax* to argmax as $\to \infty$ for all vectors that have a unique maximum value, we can establish pointwise convergence of objective in Equation 2 to objective in Equation 1 as $\to \infty$, as long as there are no ties among the top-k scores of the beam expansion candidates at any time step. We posit that absolute ties are unlikely due to random initialization of weights and the domain of the scores being $\mathbb{R}$. Empirically, we did not observe any noticeable impact of potential ties on the training procedure and our approach performed well on the tasks as discussed

in Section 4.6.

$$G_{\mathrm{DL};\ }(\mathbf{x};\ ;\mathbf{y}\ ) \xrightarrow[p]{!\ 1} G_{\mathrm{DL}}(\mathbf{x};\ ;\mathbf{y}\ ) \tag{4.3}$$

We experimented with different annealing schedules for  starting with *non-peaked softmax* moving toward *peaked-softmax* across epochs so that learning is stable with informative gradients. This is important because cost functions like Hamming distance with very high  tend to be non-smooth and are generally flat in regions far away from changepoints and have a very large gradient near the changepoints which makes optimization difficult.

### 4.3.4 Decoding

The motivation behind our approach is to make the optimization aware of beam search decoding while maintaining the continuity of the objective. However, since our approach doesn't introduce any new model parameters and optimization is agnostic to the architecture of the seq2seq model, we were able to experiment with various decoding schemes like locally normalized greedy decoding, and hard beam search, once the model has been trained.

However, to reduce the gap between the training procedure and test procedure, we also experimented with *soft beam search decoding*. This decoding approach closely follows Algorithm 3, but along with soft back pointers, we also compute hard back pointers at each time step. After computing all the relevant quantities like model score, loss etc., we follow the *hard* backpointers to obtain the best sequence $\hat{\mathbf{y}}$. This is very different from hard beam decoding because at each time step, the selection decisions are made via our soft continuous relaxation which influences the scores, LSTM hidden states and input embeddings at subsequent time-steps. The hard backpointers are essentially the MAP estimate of the soft backpointers at each step. With small, finite , we observe differences between soft beam search and hard beam search decoding in our experiments.

## 4.4 Comparison with Max-Margin Objectives

Max-margin based objectives are typically motivated as another kind of surrogate training objective which avoid the discontinuities associated with direct loss optimization. Hinge loss for structured prediction typically takes the form:

$$G_{hinge} = \max(0; \max_{\mathbf{y} \in \mathcal{Y}}( \quad (\mathbf{y}; \mathbf{y}^{*}) + s(\mathbf{y})) - s(\mathbf{y}^{*}))$$

where $\mathbf{x}$ is the input sequence, $\mathbf{y}^{*}$ is the gold target sequence, $\mathcal{Y}$ is the output search space and $(\mathbf{y}; \mathbf{y}^{*})$ is the discontinuous cost function which we assume is decomposable across the time-steps of a sequence. Finding the cost augmented maximum score is generally difficult in large structured models and often involves searching over the output space and computing the approximate cost augmented maximal output sequence and the score associated with it via beam search. This procedure introduces discontinuities in the training procedure of structured max-margin objectives and renders it non amenable to training via backpropagation. Related work (Wiseman and Rush, 2016) on incorporating beam search into the training of neural sequence models does involve cost-augmented max-margin loss but it relies on discontinuous beam search forward passes and an explicit mechanism to ensure that the gold sequence stays in the beam during training, and hence does not involve back propagation through the beam search procedure itself.

Our continuous approximation to beam search can very easily be modified to compute an approximation to the structured hinge loss so that it can be trained via backpropagation if the cost function is decomposable across time-steps. In Algorithm 3, we only need to modify line 5 as:

$$s_t[i; w] \leftarrow s_{t;i} + d(w) + f(h_{t;i}; w)$$

and instead of computing $L$ in Algorithm 3, we first compute the cost augmented maximum score

as:

$$s_{max} = \textit{peaked-softmax}\ (s_T) \cdot s_T$$

and also compute the target score $s(y)$ by simply running the forward pass of the LSTM decoder over the gold target sequence. The continuous approximation to the hinge loss to be optimized is then: $G_{\text{hinge};} = \max(0; s_{max} - s(y))$. We empirically compare this approach with the proposed approach to optimize direct loss in experiments.

## 4.5 Experimental Setup

Since our goal is to investigate the efficacy of our approach for training generic seq2seq models, we perform experiments on two NLP tagging tasks with very different characteristics and output search spaces: Named Entity Recognition (NER) and CCG supertagging. While seq2seq models are appropriate for CCG supertagging task because of the long-range correlations between the sequential output elements and a large search space, they are not ideal for NER which has a considerably smaller search space and weaker correlations between predictions at subsequent time steps. In our experiments, we observe improvements from our approach on *both* of the tasks. We use a seq2seq model with a bi-directional LSTM encoder (1 layer with tanh activation function) for the input sequence $x$, and an LSTM decoder (1 layer with tanh activation function) with a fixed attention mechanism that deterministically attends to the $i$-th input token when decoding the $i$-th output, and hence does not involve learning of any attention parameters. Since, computational complexity of our approach for optimization scales linearly with beam size for each instance, it is impractical to use very large beam sizes for training. Hence, beam size for all the beam search based experiments was set to 3 which resulted in improvements on both the tasks as discussed in the results. For both tasks, the direct loss function was the *Hamming distance cost* which aims to maximize word level

accuracy.

### 4.5.1 Named Entity Recognition

For named entity recognition, we use the CONLL 2003 shared task data (Tjong Kim Sang and De Meulder, 2003) for German language and use the provided data splits. We perform no preprocessing on the data. The output vocabulary size (label space) is 10. A peculiar characteristic of this problem is that the training data is naturally skewed toward one default label ('O') because sentences typically do not contain many named entities and the evaluation focuses on the performance recognizing entities. Therefore, we modify the Hamming cost such that incorrect prediction of 'O' is doubly penalized compared to other incorrect predictions. We use the hidden layers of size 64 and label embeddings of size 8. As mentioned earlier, seq2seq models are not an ideal choice for NER (tag-level correlations are short-ranged in NER – the unnecessary expressivity of full seq2seq models over simple encoder-classifier neural models makes training harder). However, we wanted to evaluate the effectiveness of our approach on different instantiations of seq2seq models.

### 4.5.2 CCG Supertagging

We used the standard splits of CCG bank (Hockenmaier and Steedman, 2002) for training, development, and testing. The label space of supertags is 1,284 which is much larger than NER. The distribution of supertags in the training data exhibits a long tail because these supertags encode specific syntactic information about the words' usage. The supertag labels are correlated with each other and many tags encode similar information about the syntax. Moreover, this task is sensitive to the long range sequential decisions and search effects because of how it holistically encodes the syntax of the entire sentence. We perform minor preprocessing on the data similar to the preprocessing in Vaswani et al. (2016). For this task, we used hidden layers of size 512 and the supertag label embeddings were also of size 512. The standard evaluation metric for this task is the

word level label accuracy which directly corresponds to Hamming loss.

### 4.5.3 Hyperparameter tuning

For tuning all the hyperparameters related to optimization we trained our models for 50 epochs and picked the models with the best performance on the development set. We also ran multiple random restarts for all the systems evaluated to account for performance variance across randomly started runs. We pretrained all our models with standard cross entropy training which was important for stable optimization of the non convex neural objective with a large parameter search space. This warm starting is a common practice in prior work on complex neural models (Ranzato et al., 2016; Rush et al., 2015; Bengio et al., 2015).

| Training procedure | Greedy | | Hard Beam Search | | Soft Beam Search | |
|---|---|---|---|---|---|---|
| | Dev | Test | Dev | Test | Dev | Test |
| Baseline CE | 80.15 | 80.35 | 82.17 | 82.42 | 81.62 | 82.00 |
| $G_{\text{hinge};}$ annealed | - | - | 83.03 | 83.54 | 82.82 | 83.05 |
| $G_{\text{hinge};}$ =1.0 | - | - | 83.02 | 83.36 | 82.49 | 82.85 |
| $G_{\text{DL};}$ =1.0 | - | - | 83.23 | 82.65 | 82.58 | 82.82 |
| $G_{\text{DL};}$ annealed | - | - | **85.69** | **85.82** | 85.58 | 85.78 |

**Table 4.1:** Results on CCG Supertagging. Tag-level accuracy is reported in this table which is a standard evaluation metric for supertagging.

| Training procedure | CE Greedy | | Hard Beam Search | | Soft Beam Search | |
|---|---|---|---|---|---|---|
| | Dev | Test | Dev | Test | Dev | Test |
| Baseline CE | 50.21 | 54.92 | 46.22 | 51.34 | 47.50 | 52.78 |
| $G_{\text{hinge};}$ annealed | - | - | 41.10 | 45.98 | 41.24 | 46.34 |
| $G_{\text{hinge};}$ =1.0 | - | - | 40.09 | 44.67 | 39.67 | 43.82 |
| $G_{\text{DL};}$ =1.0 | - | - | 49.88 | 54.08 | 50.73 | 54.77 |
| $G_{\text{DL};}$ annealed | - | - | 51.86 | 56.15 | **51.96** | **56.38** |

**Table 4.2:** Results on Named Entity Recognition. Macro F1 over the prediction of different named entities is reported that is a standard evaluation metric for this task.

### 4.5.4 Comparison

We report performance on validation and test sets for both the tasks in Tables 4.1 and 4.2. The baseline model is a cross entropy trained seq2seq model (Baseline CE) which is also used to warm start the the proposed optimization procedures in this paper. This baseline has been compared against the approximate direct loss training objective (Section 4.3.3), referred to as $G_{DL;}$ in the tables, and the approximate max-margin training objective (Section 4.4), referred to as $G_{hinge;}$ in the tables. Results are reported for models when trained with annealing , and also with a constant setting of = 1:0 which is a very smooth but inaccurate approximation of the original direct loss that we aim to optimize[2]. Comparisons have been made on the basis of performance of the models under different decoding paradigms (represented as different column in the tables): locally normalized decoding (CE greedy), hard beam search decoding and soft beam search decoding described in Section 4.3.4.

## 4.6 Results

As shown in Tables 4.1 and 4.2, our approach $G_{DL;}$ shows significant improvements over the locally normalized CE baseline with greedy decoding for both the tasks (+5.5 accuracy points gain for supertagging and +1.5 F1 points for NER). The improvement is more pronounced on the supertagging task, which is not surprising because: (i) the evaluation metric is tag-level accuracy which is congruent with the Hamming loss that $G_{DL;}$ directly optimizes and (ii) the supertagging task itself is very sensitive to the search procedure because tags across time-steps tend to exhibit long range dependencies as they encode specialized syntactic information about word usage in the sentence.

Another common trend to observe is that annealing always results in better performance than training with a constant = 1:0 for both $G_{DL;}$ (Section 4.3.3) and $G_{hinge;}$ (Section 4.4).

[2]Our pilot experiments that involved training with a very large constant resulted in unstable optimization.

This shows that a stable training scheme that smoothly approaches minimizing the actual direct loss is important for our proposed approach. Additionally, we did not observe a large difference when our soft approximation is used for decoding (Section 4.3.4) compared to hard beam search decoding, which suggests that our approximation to the hard beam search is as effective as its discrete counterpart.

For supertagging, we observe that the baseline cross entropy trained model improves its predictions with beam search decoding compared to greedy decoding by 2 accuracy points, which suggests that beam search is already helpful for this task, even without search-aware training. Both the optimization schemes proposed in this paper improve upon the baseline with soft direct loss optimization ($G_{\mathrm{DL};\ }$), performing better than the approximate max-margin approach. [3]

For NER, we observe that optimizing $G_{\mathrm{DL};\ }$ outperforms all the other approaches but we also observe interesting behaviour of beam search decoding and the approximate max-margin objective for this task. The pretrained CE baseline model yields worse performance when beam search is done instead of greedy locally normalized decoding. This is because the training data is heavily skewed toward the 'O' label and hence the absolute score resolution between different tags at each time-step during decoding isn't enough to avoid leading beam search toward a wrong hypothesis path. We observed in our experiments that hard beam search resulted in predicting more 'O's which also hurt the prediction of tags at future time steps and hurt precision as well as recall. Encouragingly, $G_{\mathrm{DL};}$ optimization, even though warm started with a CE trained model that performs worse with beam search, led to the NER model becoming more search aware, which resulted in superior performance. However, we also observe that the approximate max-margin approach ($G_{\mathrm{hinge};\ }$) performs poorly here. We attribute this to a deficiency in the max-margin objective when coupled with approximate search methods like beam search that do not provide guarantees on finding the supremum: one way

[3] Separately, we also ran experiments with a max-margin objective that used hard beam search to compute loss-augmented decodes. This objective is discontinuous, but we evaluated the performance of gradient optimization nonetheless. While not included in the result tables, we found that this approach was unstable and considerably underperformed both approximate max-margin and direct loss objectives.

to drive this objective down is to learn model scores such that the search for the best hypothesis is difficult, so that the value of the loss augmented decode is low, while the gold sequence maintains higher model score. Because we also warm started with a pre-trained model that results in a worse performance with beam search decode than with greedy decode, we observe the adverse effect of this deficiency. The result is a model that scores the gold hypothesis highly, but yields poor decoding outputs. This observation indicates that using max-margin based objectives with beam search during *training* actually may achieve the opposite of our original intent: the objective can be driven down by *introducing* search errors.

The observation that our optimization method led to improvements on *both* the tasks–even on NER for which hard beam search during decoding on a CE trained model hurt the performance–by making the optimization more search aware, indicates the effectiveness of our approach for training seq2seq models.

## 4.7 Conclusion

While beam search is a method of choice for performing search in neural sequence models, as our experiments confirm, it is not necessarily guaranteed to improve accuracy when applied to cross-entropy-trained models. We proposed a novel method for optimizing model parameters that directly takes into account the process of beam search itself through a continuous, end-to-end sub-differentiable relaxation of beam search composed with the final evaluation loss. Experiments demonstrate that our method is able to improve overall test-time results for models using beam search as a test-time inference method, leading to substantial improvements in accuracy.

As discussed earlier, another closely related work to incorporating beam search into training of sequence models is work by Wiseman and Rush (2016) (BSO) which relies on discontinuous beam search forward passes and involves a variant of cost-augmented max-margin loss and an explicit loss to ensure that the gold sequence stays in the beam during training. This training scheme and

loss function was found to be effective for tasks like Machine Translation and it would be interesting to study the difference between this training strategy that focuses on designing better loss functions and the training strategy described in our work that focuses on noisy gradients of the continuous relaxation to beam search.

# Chapter 5

# Globally Normalized Sequence Models: Ameliorating Label Bias

This chapter explores the advantages that global normalization provides over local normalization. In particular, the notion of label bias as a harmful inductive bias for training autoregressive models is described in details. As mentioned in Chpter 1, the step-wise normalization constraint in autoregressive models is the primary reason for poor training of these models which addles these models with unesirable behavior like inability to recover from early decoded prefixes, ignoring the input context for conditional generation etc. In this chapter, I identify the two major ways in which *label bias* manifests itself inn autoregressive models. This chapter then proposes a beam-search aware training algorithm for globally normalized models that helps ameliorate label bias.

## 5.1 Label Bias

Locally normalized models are often associated with *label bias* because they conserve the probability mass (Bottou, 1991) entering and exiting each state that associated with incremental decoding with autoregressive models. This causes autoregressive models to be extremely sensitive to deviations of

autoregressively parametrized softmaxes from the true underlying distribution during decoding often leading to undesirable behaviors like inability to revise previously made decisions, ignoring the input during conditional generation, and assigning high likelihoods to bad sequences. In this thesis, I use the term *label bias* to refer to this negative inductive bias arising due to the normalization constraint in autoregressive models that is responsible for undesirable behavior typically demonstrated by seemingly well-trained autoregressive models.

As discussed in section 2.5, different global parametrizations can correspond to same autoregressive distributions which suggests that mere magnitude differences among the energy/scores of different globally normalized models with the same autoregressive distributions will yield different max-weighted strings with approximate search algorithms. As discussed later, this explains the differences in the training and decoding behaviors between autoregressive and globally noermalized models. More strikingly, while slight miscalibrations under uncertain context would derail well-trained autoregressive models, some globally normalized parametrizations among all the parametrizations representing the autoregressive distribution might be better suited to recover from miscalibration errors.

In this thesis, we focus on conditional generation, and describe two major contexts in which undesirable aspects of label bias are manifested: i) conditioning on partial input, and ii) inexact inference. The first condition has been studied in great detail in the context of comparing the locally normalized maximum entropy Markov models(MEMMs) with the globally normalized conditional random fields(CRFs). This condition is most commonly used to illustrate the ill effects of label bias and hence, the exposition of label bias have largely been restricted to this setting. However, this common setting of access to partial input is largely eliminated by expressive neural encoders that are capable of representing information about the entire input. In this thesis, we discover and describe an alternate manifestation of label bias that is more relevant to the contemporary powerful neural model classes for sequences, which is emirically apparent in poor performance of autoregressive models compared to their globally normalized conterparts when trained with a search-aware training

objective.

### 5.1.1 Label Bias and access to partial input

It was shown in (Andor et al., 2016; Lafferty et al., 2001), locally normalized conditional models *with access to only partial input*, $x_{1:i-1}$, at each decoding step are biased towards labeling decisions with low-entropy transition probabilities at each decoding step and, as a result, suffer from a weakened ability to revise previous decisions based upon future input observations. This phenomenon has been referred to as *label bias* inn literature, and presents itself as an arbitrary allocation of probability mass to unlikely or undesirable label sequences despite the presence of well-formed sequences in training data. Andor et al. (2016) prove that this class of locally normalized models that relies on the structural assumption of access to only left-to-right partial input at each step,

$$\prod_{i=1}^{T} p(y_i \mid \mathbf{x}; y_{1:i-1}) = \prod_{i=1}^{T} p(y_i \mid x_{1:i-1}; y_{1:i-1});$$

is strictly less expressive than its globally normalized counterpart.

However, the standard sequence-to-sequence models used most often in practice and presented in this paper actually condition the decoder on a summary representation of the *entire input sequence*, $\mathbf{x}$, computed by a neural encoder. Hence, depending on the power of the encoder, it is commonly thought that such models avoid this type of label bias. For these models, both locally normalized and globally normalized conditional models are equally expressive, in principle, with a sufficiently powerful encoder.

As we suggest in the next section, and show empirically in experiments, this does not necessarily mean that both autoregressive and globally normalized parametrizations are equally amenable to gradient-based training in practice, particularly when the search space is large and search-aware training techniques are used.

### 5.1.2 Label Bias and inexact search

To improve performance with inexact decoding methods (e.g. beam search), search-aware training techniques take into account the decoding procedure that will be used at test time and adjust the parameters of the model to maximize prediction accuracy under the decoder. Because of the popularity of beam search as a decoding procedure for sequence models, we focus on beam search-aware training. While many options are available, including beam-search optimization (BSO) (Wiseman and Rush, 2016), we use search-aware training strategy based upon continuous relaxation to beam search described in chapter 4.

We illustrate via example how optimization of locally normalized models may suffer from a new manifestation of label bias when using beam search-aware training, and point to reasons why this issue might be mitigated by the use of globally normalized models. While the scores of successors of a single candidate under a locally normalized model are constrained to sum to one, scores of successors under a globally normalized model need only be positive. Intuitively, during training, this gives the globally normalized model more freedom to down-weight undesirable intermediate candidates in order avoid search errors.

In the example beam search decoding problem in Figure 5.1, we compare the behavior of locally and globally normalized models at a single time step for a beam size of two. In this example, we assume that the score for beams in both the models is exactly the same until the step shown in Figure 5.1. Suppose that the lower item on the beam (X2) is correct, and thus, for more effective search, we would prefer the models scores to be such that only successors of the lower beam item are present on the beam at the next step. However, since, the scores at each step for a locally normalized model are constrained to sum to one, the upper beam item (X1) generates successors with scores comparable to those of the lower beam item. As we see in the example, due to the normalization constraint, search-aware training of the locally normalized model might find it difficult to set the parameters to prevent extension of the poorer candidate. In contrast, because the scores of a *globally normalized* model are not constrained to sum to one, the parameters of the neural model can be set

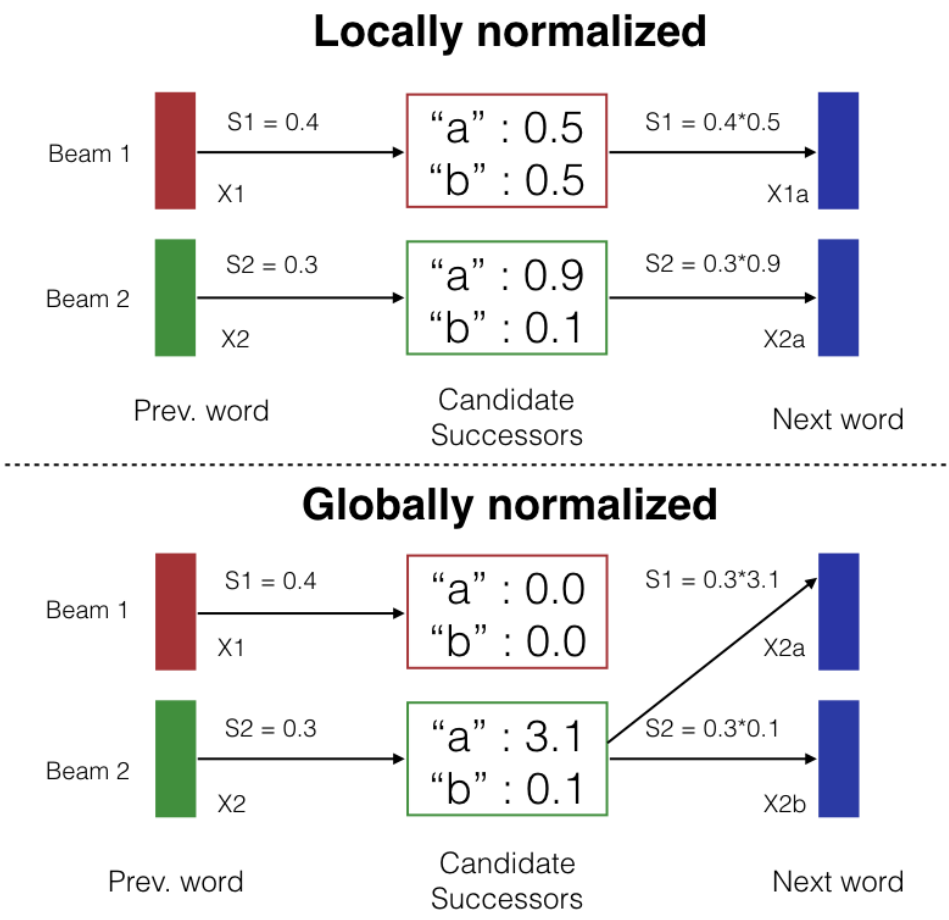


**Figure 5.1:** Illustrative example of bias arising in locally normalized models due to beam search. Red indicates the candidate that optimization should learn to discard and green indicates the candidate that should be propagated. Locally normalized models are constrained to return normalized scores for the successors of each candidate, while globally normalized models are unconstrained and can more easily learn to drop successors of the red candidate.

such that *all* the successors of the bad candidate have a very low score and thus do not compete for space on the beam. This illustrates a mechanism by which search-aware training of globally normalized models in a large search spaces might be more effective. However as discussed earlier, if we can perform *exact search* then this *label bias* ceases to exist because both the models have the same expressive power with a search-agnostic optimization scheme. In experiments, we will explore this trade-off empirically.

## 5.2 Search-aware Training for Globally Normalized Models

In this section, we investigate the effectiveness of globally normalized sequence models at ameliorating label bias when compared to similarly parametrized locally normalized models. We extend the previously described approach for search-aware training via a continuous relaxation of beam search (Goyal et al., 2018) in order to enable training of *globally normalized* recurrent sequence models through simple backpropagation. (Smith and Johnson, 2007) proved that locally normalized conditional PCFGs and unnormalized conditional WCFGs are equally expressive for finite length sequences and posit that Maximum Entropy Markov Models (MEMMs) are weaker than CRFs because of the structural assumptions involved with MEMMs that result in label bias. We find out in our experiment that in spite of having the same expressive power, training globally normalized models is easier and results in better models, when search-aware training is done in the context of huge search space which rules out exact search and non-convex optimization. For controlled comparison, we experiment with the same underlying neural architecture for training both locally and globally normalized models in a search-aware manner. We then use the extension to the beam search aware training to conduct an empirical study of the interaction between global normalization, high-capacity encoders, and search-aware optimization. In particular, a major modification is that while for training search-aware locally normalized models via continuous relaxation to beam search, we use *log-normalized* successor scores at each decode step, for training globally normalized models however, we directly use *unnormalized scores* at each time step, which are $\in \mathsf{R}_+$. This has an effect of associating a non-negative score with each sequence in the finite hypothesis space which has to be explicitly normalized in order to get the probability of the sequence. Since, a probabilistic formulation is not necessary for beam search to work, beam-search aware training allows us to train *unnormalized* models.

However, we note that our proposed approach is not the only approach to train globally normalized sequence models. Most other alternatives for search-aware training of globally normalized

neural sequence models include approaches that use some mechanism like early updates (Collins and Roark, 2004) that relies on explicitly tracking if the gold sequence falls off the beam and is not end-to-end continuous. (Andor et al., 2016) describe a method for training globally normalized neural feedforward models, which involves optimizing a CRF-based likelihood where the normalizer is approximated by the sum of the scores of the final beam elements. They describe and focus on label bias arising out of conditioning on partial input and hence focused on the scenario in which locally normalized models can be less expressive than globally normalized models, whereas we also consider another source of label bias which might be affecting the optimization of equally expressive locally and globally normalized conditional models. (Wiseman and Rush, 2016) also propose a beam search based training procedure that uses unnormalized scores similar to our approach. Their models achieve good performance over CE baselines – a pattern that we observe in our results as well. Our approach to training unnormalized models however is end-to-end sub-differentiable and the whole training procedure can be encoded via a computation graph in an autodiff library. Our main focus is to empirically analyze the factors affecting the boost in performance with end-to-end continuous search-aware training (Goyal et al., 2018) for globally normalized models. We observe that in the context of inexact search, globally normalized neural models are still more effective than their locally normalized counterparts.

Further, since our training approach is sensitive to warm-starting with pre-trained models, we also propose a novel initialization strategy based on self-normalization for pre-training globally normalized models.

## 5.3 Initialization for training globally normalized models

Goyal et al. (2018) reported that *initialization* with a locally normalized model pre-trained with teacher-forcing was important for their continuous beam search based approach to be stable and hence they used the locally normalized log-scores for their search-aware training model. In this

work, we experimented with the unnormalized candidate successor scores and found that initializing the optimization for a globally normalized objective with a cross-entropy trained locally normalized model resulted in unstable training. This is expected because the locally normalized models are parametrized in a way such that using the scores before the softmax normalization results in a very different outcome than using scores after local normalization. For example, the locally normalized Machine Translation model in Table 5.1, that gives a BLEU score of 27.62 when decoded with beam search using locally normalized scores, results in BLEU of 4.30 when beam search decoding is performed with unnormalized scores. Pre-training a truly globally normalized model for initialization is not straightforward because no exact likelihood maximization techniques exist for globally normalized models as the global normalizer is intractable to compute.

Therefore, we propose a new approach to initialization for search-aware training of globally normalized models: we pre-train a locally normalized model that is parametrized like a globally normalized model. More specifically, we train a locally normalized model with its distribution over the output sequences denoted by $p_L(\mathcal{Y})$ such that we can easily find a globally normalized model with a distribution $p_G(\mathcal{Y})$ that matches $p_L(\mathcal{Y})$. For a locally normalized model, the log-probability of a sequence is:

$$\sum_{i=1}^{} [\log s(\mathbf{x}; y_{1:i-1}; y_i) - \log Z_{L;i}(\mathbf{x}; y_{1:i-1})]$$

and for a globally normalized model it is:

$$\left[ \sum_{i=1}^{} \log s(\mathbf{x}; y_{1:i-1}; y_i) \right] - \log Z_G(\mathbf{x})$$

### 5.3.1 Self Normalization

One way to find a locally normalized model that is parametrized like a globally normalized model is to ensure that the local normalizer at each step, $\log Z_{L;i}(\mathbf{x}; y_{1:i-1})$, is 0. With the local normalizer

being zero it is straightforward to see that the log probability of a sequence under a locally normalized model can easily be interpreted as log probability of the sequence under a globally normalized model with the global log-normalizer, $\log Z_G(\mathbf{x}) = 0$. This training technique is called *self-normalization* (Andreas and Klein, 2015) because the resulting models' unnormalized score at each step lies on a probability simplex. A common technique for training self-normalized models is L2-regularization of local log normalizer which encourages learning a model with $\log Z = 0$ and was found to be effective for learning a language model by Devlin et al. (2014)[1]. The L2-regularized cross entropy objective is given by:

$$\min \sum_{\mathbf{x},\mathbf{y} \in D} \sum_{i=1}^{|\mathbf{y}|} -\log p(y_i \mid \mathbf{x}, y_{1:i-1}) + \alpha \cdot (\log Z_{L,i}(\mathbf{x}, y_{1:i-1}))^2$$

In Table 5.1, we report the mean and variance of the local log normalizer on the two different tasks using L2-regularization (*L2*) based self normalization and no self normalization (*CE*). We observe that *L2* models are competitive performance-wise to the cross-entropy trained locally normalized models while resulting in a much smaller local log-normalizer on average. Although, we couldn't minimize $\log Z$ exactly to 0, we observe in Section 5.5 that this is sufficient to train a reasonable initializer for the search-aware optimization of globally normalized models. It is important to note that these approaches yield a *globally normalized* model that is equivalent to a locally normalized model trained via teacher-forcing and hence these are only used to *warm-start* the search-aware optimization of globally normalized models. Our search-aware training approach is free to adjust the parameters of the models such that the final globally normalized model has a non-zero log-normalizer $Z_G$ over the data.

Other possible approaches to project locally normalized models onto globally normalized models

[1]Noise Contrastive Estimation (Mnih and Teh, 2012; Gutmann and Hyvärinen, 2010) is also an alternative to train unnormalized models but our experiments with NCE were unstable and resulted in worse models.

| | | Train logZ | | Dev logZ | | Acc/ |
|---|---|---|---|---|---|---|
| | | Mean | Var | Mean | Var | BLEU |
| CCG | CE | 21.08 | 9.57 | 21.96 | 9.18 | 93.3 |
| | L2 | 0.6 | 0.29 | 0.26 | 0.08 | 91.9 |
| MT | CE | 24.7 | 115.4 | 25.8 | 129.1 | 27.62 |
| | L2 | 0.65 | 0.18 | 0.7 | 0.29 | 26.63 |

**Table 5.1:** Comparison of logZ between cross entropy trained models (CE) and self normalized models (L2) for CCG supertagging and Machine Translation tasks.

include distribution matching via knowledge distillation (Hinton et al., 2015). We leave exploration of warm-starting of search aware optimization with this approach to future work.

## 5.4 Experimental setup

To empirically analyze the interaction between label bias arising from different sources, search-aware training, and global normalization, we conducted experiments on two tasks with vastly different sizes of output space: CCG supertagging and Machine Translation. As described in the next section, the task of tagging allows us to perform controlled experiments which explicitly study the effect of amount of input information available to the decoder at each step, we analyze the scenarios in which search aware training and global normalization are expected to improve the model performance.

In all our experiments, we report results on training with standard teacher forcing optimization and self-normalization as our baselines. We report results with both search-aware locally and globally normalized models after warm starting with both cross entropy trained models and self-normalized models to study the effects of search-aware optimization and global normalization. We follow Goyal et al. (2018) and use the decomposable Hamming loss approximation with search-aware optimization for both the tasks and decode via *soft beam search decoding* method which involves continuous beam search with soft backpointers for the LSTM Beam search dynamics as described in chapter 4, but using identifiable backpointers and labels (using MAP estimates of soft

backpointers and labels) to decode.

We tune hyperparameters like learning rate and annealing schedule by observing performance on development sets for both the tasks. We performed at least three random restarts for each class and report results based on best development performance.

### 5.4.1 CCG supertagging

We used the standard splits of CCG bank (Hockenmaier and Steedman, 2002) for training, development, and testing. The label space of supertags is 1,284 and the labels are correlated with each other based on their syntactic relations. The distribution of supertag labels in the training data exhibits a long tail distribution. This task is sensitive to the long range sequential decisions because it encodes rich syntactic information about the sentence. Hence, this task is ideal to analyze the effects of label bias and search effects. We perform minor preprocessing on the data similar to the preprocessing in Vaswani et al. (2016). For experiments related to search aware optimization, we report results with beam size of 5.[2]

### 5.4.2 Tagging model for ablation study

We changed the standard sequence-to-sequence model to be more suitable for the tagging task. This change also lets us perform controlled experiments pertaining to the amount of input sequence information available to the decoder at each time step.

In a standard encoder-decoder model with attention, the initial hidden state of the decoder is often some function of the final encoder state so that the decoder's predictions can be conditioned on the full input. For our tagging experiments, instead of influencing the initial decoder state with the encoder, we set it to a vector of zeros. Thus the information about input for prediction is *only* available via the attention mechanism. In addition to the change above, we also forced the model to

[2]We observed similar results with beam size 10

attend to only the $i^{th}$ input representation while predicting the $i^{th}$ label. This is enforceable because the output length is equal to the input length and it is also a more suitable structure for a tagging model. With these changes in the decoder, we can precisely control the amount of information about the input available to the decoder at each prediction step. For example, with a unidirectional LSTM encoder, the decoder at $i^{th}$ step only has access to input till the $i^{th}$ token and the prediction history:

$$p(y_i \mid \mathbf{x}; y_{1:i-1}) = p(y_i \mid x_{1:i}; y_{1:i-1})$$

This setting lets us clearly explore the classical notion of *label bias* arising out of access to partial input at each prediction step (Section 5.1.1). A bidirectional LSTM encoder, however provides access to all of the input information to the decoder at all the prediction steps.

| | Unidirectional | Bidirectional |
|---|---|---|
| pretrain-greedy | 76.54 | 92.59 |
| pretrain-beam | 77.76 | 93.29 |
| locally normalized | 83.9 | **93.76** |
| globally normalized | **83.93** | 93.73 |

**Table 5.2: Accuracy results on CCG supertagging when initialized with a regular teacher-forcing model**. Reported using *Unidirectional* and *Bidirectional* encoders respectively with fixed attention tagging decoder. *pretrain-greedy* and *pretrain-beam* refer to the output of decoding the initializer model. *locally normalized* and *globally normalized* refer to search-aware soft-beam models

| | Unidirectional | Bidirectional |
|---|---|---|
| pretrain-greedy | 73.12 | 91.23 |
| pretrain-beam | 73.83 | 91.94 |
| locally normalized | 83.35 | **92.78** |
| globally normalized | **85.50** | 92.63 |

**Table 5.3: Accuracy results on CCG supertagging when initialized with a self normalized model**.

| Init-scheme → | Regular | Self-normalized |
|---|---|---|
| pretrain-greedy | 26.24 | 25.42 |
| pretrain-beam | 27.62 | 26.63 |
| locally-normalized | **29.28** | 27.71 |
| globally-normalized | 26.24 | **29.27** |

**Table 5.4: BLEU results on de-en Machine Translation.** *Regular* and *Self-normalized* refer to the initization scheme for soft beam search training. *pretrain-greedy* and *pretrain-beam* refer to the output of decoding the initializer model. *locally normalized* and *globally normalized* refer to search-aware soft-beam models

### 5.4.3 Machine Translation

We use the same dataset (the German-English portion of the IWSLT 2014 machine translation evaluation campaign (Cettolo et al., 2014)), preprocessing and data splits as Ranzato et al. (2016) for our Machine Translation experiments. The output label/vocabulary size is 32000 and unlike tagging, the length of output sequences cannot be deterministically determined from the length of the input sequence. Moreover, the output sequence does not necessarily align monotonically with the input sequence. Hence the output sequence space for MT is much larger than that for tagging and the effects of inexact search on optimization are expected to be even more apparent for MT. We use a standard LSTM-based encoder/decoder model with a standard attention mechanism (Bahdanau et al., 2015) for our MT experiments. For search-aware optimization experiments, we report results with beam size 3.[3]

## 5.5 Results and Analysis

The results reported in Tables 5.2, 5.3 and 5.4 allow us to analyze the effect of interaction of label bias, inexact search and global normalization in detail.

[3]We observed similar results beam size of 5.

### 5.5.1 Label bias with partial input

First, we analyze the effect of label bias that arises from conditioning on partial input (Section 5.1.1) during decoding on optimization of the models. The unidirectional encoder based tagging experiments suggest that conditioning on partial input during decoding results in poor models when trained with cross entropy based methods. Interestingly, all techniques improve upon this: (i) search-aware locally and globally normalized models are able to train for accuracy directly and eliminate exposure bias that arises out of the mismatch between train-time and test-time prediction methods, and, (ii) the bidirectional tagging model which provides access to all of the input is powerful enough to learn a complex relationship between the decoder and the input representations for the search space of the CCG supertagging task and results in a much better performance.

### 5.5.2 Initialization of search-aware training

Next, we analyze the importance of appropriate initialization of search-aware optimization with pretrained models. Across all the results in Tables 5.2, 5.3 and 5.4, we observe that search-aware optimization for locally normalized models always improves upon the pre-trained locally normalized models used for initialization. But when the search-aware optimization for globally normalized models is initialized with locally normalized CE models, the improvement is not as pronounced and in the case of MT, the performance is actually *hurt* by the improper initialization for training globally normalized models – probably a consequence of large search space associated with MT and incompatibility between unnormalized scores for search-aware optimization and locally normalized scores of the *CE* model used for pre-training. When the *self-normalized* models are used for initialization, optimization for globally normalized models always improves upon the pre-trained self-normalized model. It is interesting to note that we see improvements for the globally normalized models even when $logZ$ is not exactly reduced to $0$ indicating that the scores used

for search-aware training initially are comparable to the scores of the pre-trained self-normalized model. We also observe that self-normalized models perform slightly worse than CE-trained models but search aware training for globally normalized models improves the performance significantly.

### 5.5.3 Search-aware training

Next, we analyze the effect of search-aware optimization on the performance of the models. Search-aware training with locally normalized models improves the performance significantly in *all* our experiments which indicates that accounting for exposure bias and optimizing for predictive performance directly is important. We also observe that the bidirectional model for tagging is quite powerful and seems to account for both *exposure bias* and *label bias* to a large extent. We reckon that this may be because the greedy decoding itself is very close to *exact search* for this well-trained tagging model over a search space that is much simpler than that associated with MT. Therefore, the impact of search-aware optimization on the bidirectional tagger is marginal. However, it is much more pronounced on the task of MT.

### 5.5.4 Global normalization and label bias

We analyze the importance of training globally normalized models. In the specific setup for tagging with the unidirectional encoder, globally normalized models are actually *more expressive* than the locally normalized models (Andor et al., 2016) as described in Section 5.1.1 and this is reflected in our experiments (table 5.3) with tagging. The globally normalized model (warm started with a self-normalized model) performs the best among all the models in the unidirectional tagger case which indicates that it is ameliorating something beyond exposure bias which is fixed by the search-aware locally normalized model.

For MT (table 5.4), both globally normalized and locally normalized models are equally expressive in theory because the decoder is conditioned on the full input information at each step, but we still observe that the globally normalized model improves significantly over the self-normalized pre-trained model and the search-aware locally normalized model. This indicates that it might be ameliorating the label bias associated with inexact search (discussed in Section 5.1.2). As discussed in Section 5.3, the globally normalized model, when initialized with a CE trained model, performs worse because of improper initialization of the search aware training. The self-normalized model starts off 1 BLEU point worse than the CE model point but global normalization, initialized with the self-normalized model improves the performance and is competitive with the best model for MT. This suggests that a better technique for initializing the optimization for globally normalized models should be helpful in improving the performance.

### 5.5.5 Global normalization and sentence length

| | N-gram overlap | Length ratio |
|---|---|---|
| pretrain-beam | 63.5/35.7/21.8/13.7 | 0.931 |
| locally-normalized | 66.9/39.4/22.7/14.0 | 0.918 |
| globally-normalized | 65.0/39.1/23.2/14.7 | 0.959 |

**Table 5.5: Breakdown of BLEU results on de-en Machine Translation dev set.** Reported on Self-normalized initialization

| Src sent-length → | **0-20** | **20-30** | **30-40** | **40+** |
|---|---|---|---|---|
| pretrain-beam | 29.36 | 25.73 | 24.71 | 24.50 |
| locally-normalized | 32.35 | 26.95 | 25.39 | 25.2 |
| globally-normalized | 33.21 | 28.08 | 26.75 | 26.41 |

**Table 5.6: BLEU scores with different length inputs on dev set** Reported on Self-normalized initialization. The header specifies the range of length of the input sentences

In tables 5.5 and 5.6, we analyze the source of improvement from global normalization for MT. In table 5.5, we report the ngram overlap scores and ratio of length of the predictions to length

of hypothesis for the case when the search-aware training is initialized with a self-normalized model. We observe that the globally normalized model produces longer predictions than the locally normalized model. More interestingly, it seems to have better 3 and 4-gram overlap and slightly worse unigram and bigram overlap score than the locally normalized model. These observations suggest that globally normalized models are better able to take longer range effects into account and are also cautious about predicting the *end-of-sentence* symbol too soon. Moreover, in table 5.6, we observe that globally normalized models perform better on all the length ranges but especially so on long sentences.

# Chapter 6

# Energy Based Neural Sequence Models: Connection with Masked Language Modelling objective

## 6.1 Introduction

As described in **Chapter 2**, globally normalized sequence models whose support contains a finite but large set of discrete sequences are easily rewritten as energy-based sequence models which assign a scalar score to each sentence and whose defined probability distribution is obtained by summation of scores of all the possible sequences. This part of the thesis focuses on training energy-based models for natural language sequences. Training such models is difficult and finicky, therefore, this work (Goyal et al., 2021) mainly aims to interpret the well-behaved but poorly understood masked language objective as being related to implicit optimization of an effective energy-based model over sequences. The effectiveness of the MLM objective is studied by comparing energy-based models trained on synthetic data with enumerable search spaces via the MLM objective, direct maximization of pseudo log likelihood and direct maximization of log likelihood.

Masked language modeling (MLM) objectives (Devlin et al., 2019; Yang et al., 2019; Gu et al., 2018) for sequences, although recent, have become ubiquitous for many Natural Language Processing (NLP) applications (Liu et al., 2019; Zhang* et al., 2020; Rogers et al., 2021) because they are easy to optimize and enable learning of highly expressive and flexible representations by the virtue of conditioning on bidirectional context (Peters et al., 2018; Devlin et al., 2019). However despite their popularity, they lack a principled probabilistic interpretation and hence sampling from MLMs, or characterizing uncertainty about the predictions made with them has remained elusive. This drawback is reflected in the observation that recently proposed non-probabilistic approaches for *generating* high-scoring sequences from these MLMs (Ghazvininejad et al., 2019) still trail probabilistic autoregressive models (Brown et al., 2020; Sutskever et al., 2014) despite having access to greater bidirectional context while generating.

In this work, we posit that optimizing MLM objectives results in training of *implicit energy-based sequence models* that correspond to probability distributions over natural language sequences by assigning a score to each possible sequence in the large but finite sequence space. To explore the veracity of this claim, we *develop and experiment with two energy parametrizations* (or scoring schemes) that can be easily derived from the representations learned by the trained MLMs' transformer networks. These parametrizations have been inspired by the success of recent work on using MLMs for sentence-level judgements for discriminating between probable and improbable sequences (Salazar et al., 2020; Zhang* et al., 2020).

Although, it is easy to compute the energy/score of a sequence with these MLM-based parametrizations, the bidirectional nature of MLMs precludes efficient sampling algorithms like ancestral sampling. Therefore, a *primary contribution* of this work is to *develop Metropolis-Hastings (MH) based sampling algorithms* for these energy networks. While it is tempting to formulate a Gibbs sampling scheme (Gelfand and Smith, 1990) based on the positional masked conditional distributions used for training the MLMs (Wang and Cho, 2019), we theoretically and empirically demonstrate that these masked conditional distributions do not necessarily correspond to any joint

distribution or energy network and hence result in *invalid* Gibbs samplers. Instead, we propose to use these masked conditionals as proposal distributions for transitioning to a new state (sequence) in the Markov chain of an MCMC sampler based on the Metropolis-Hastings algorithm (Hastings, 1970). *Another contribution* of our work is to *design a block-replacement proposal distribution* for improve mixing of the Markov chain in our proposed MH sampling framework, which results in faster generation and better samples.

We empirically investigate the effectiveness of the two proposed energy parametrizations by examining the quality of samples drawn from these energy-models on the conditional generation task of Machine Translation (MT). We observe that high BLEU scores are correlated with low energy values which indicates that these parametrizations are reasonable proxies for the desired implicit bidirectional energy network trained via the MLM objective for MT. We study the behavior of our sampling approach extensively with different proposal distributions. We also verify the effectiveness of our approach by sampling from regions around the mode by annealing the target distribution and finding our *samples* to be competitive with a prominent undirected (and non-probablistic) generation approach (Ghazvininejad et al., 2019) on MT performance.

We find our proposed sampler to generate high-quality sequences (Energy-wise and BLEU-wise) under the proposed energy parametrizations which suggests that the optimization of MLM objective is implicitly equivalent to training global energy network that induces probability distribution over the space of sequences. While in this work, we primarily focus on sampling from the energy network underlying MLMs, our findings open up avenues for developing more direct, stable and simple training procedures for energy-based sequence models inspired from the MLM objectives and our proposed MH sampling scheme.

**Related work:** Gradient based training of energy networks (LeCun et al., 2006; Zhao et al., 2017; Du and Mordatch, 2019) has been successful at training models for inducing distributions over continuous domains but are not suitable for training discrete sequence models for text. To overcome this problem, recent work has proposed continuous relaxations to the discrete domain (Belanger and

McCallum, 2016; Grathwohl et al., 2021), but the unordered nature of discrete symbols in text leads to brittle and unsuccessful training. Direct training of energy networks for text tends to be expensive and unstable as well (Goyal et al., 2019; Deng et al., 2020; Tu et al., 2020; Zhang et al., 2017). While MLM objectives(Devlin et al., 2019; Clark et al., 2020a,b) in contrast, are easy to train and learn good representations of textual data, they do not have a probabilistic interprertation. In this work, we interpret MLMs as implicit energy networks and develop approaches to sample from them. While there have been attempts to generate sequences from MLMs in a non(pseudo)-probabilistic manner (Wang and Cho, 2019; Ghazvininejad et al., 2019; Gu et al., 2018; Mansimov et al., 2019), the techniques introduced in this work sample correctly from the energy networks underlying MLMs.

## 6.2 Masked Language Models and Energy Networks

We can only directly obtain the conditional distributions of the `[MASK]` tokens, conditioned on the rest of the tokens in the sequence from an MLM. In this section, we discuss potential parametrizations of energy functions that could correspond to the implicit networks trained via MLM objectives and describe how to obtain these energy values from the trained MLMs. Let $\mathcal{Y}$ be the space of all finite sequences and $p(\mathbf{y};\ )$ be the probability of the sequence $\mathbf{y} \in \mathcal{Y}$ under the target distribution defined by the energy function $E(\mathbf{y};\ )$ parametrized by , defined as follows:

$$p(\mathbf{y};\ ) = \frac{e^{\ E(\mathbf{y};\ )}}{\sum_{\mathbf{y}' \in Y} e^{\ E(\mathbf{y}';\ )}} = \frac{(\mathbf{y};\ )}{Z(\ )}$$

where represents the unnormalized score of the sequence $\mathbf{y}$ and $Z(\ )$ is the intractable normalization constant computed by summing over all the sequences.[1] We propose *two* parametrizations for the energy functions: 1) *Raw* scoring, and 2) *Locally normalized* scoring.

[1]For conditional generation, the energy function and the probability distribution over $\mathbf{y}$ is also dependent on the conditioning context $\mathbf{x}$; however, for notational convenience we suppress this explicit dependence in rest of the description.

### 6.2.1 Raw Scoring

For each position $t$ in the sequence $\mathbf{y}$ of length $T$, we associate a random variable $\mathbf{y}_t \in \mathbb{V}$ with the $t$-th token, where $\mathbb{V}$ is the vocabulary. MLMs learn a representation $h(\mathbf{y}_{nt})$, for $\mathbf{y}_t$ that is sensitive to the bidirectional surrounding context $\mathbf{y}_{nt}$. For notational convenience, we use $\mathbf{y}_{i=w;ni}$ to denote a sequence $\mathbf{y}$ with the $i$-th variable taking on the value $w$. We use such bidirectional neural parametrizations to define an energy $E_{raw}$ for $\mathbf{y}$ that corresponds to fully connected MRFs (Gibbs random fields, more precisely) as the sum of the local positional scores:

$$E_{raw}(\mathbf{y};\ ) = -\sum_{t=1}^{T} \log\ _t(\mathbf{y};\ ); \qquad \text{where } \log\ _t(\mathbf{y};\ ) = f(\mathbf{y}_t; h(\mathbf{y}_{nt}));\ )$$

**Conditional distribution under $E_{raw}$:** Performing Gibbs sampling from the MRF defined by $E_{raw}$ requires computation of this conditional distribution of a token given the surrounding context:

$$p(\mathbf{y}_i|\mathbf{y}_{ni};\ ) = \frac{\prod_t\ _t(\mathbf{y};\ )}{\sum_{w2\mathbb{V}} \prod_t\ _t(((\mathbf{y}_{i=w;ni});\ ))}$$

Computing this conditional is very expensive and would require running $|\mathbb{V}|$ passes of the MLM decoder because the positional potentials $_t$ are computed over fully connected cliques. We shortly use these true conditionals to optimize the true pseudolikelihood of an energy-based model on synthetic data for which enumeration over sequence space is possible and explore the effectiveness of doing this over optimizing via MLM conditionals for this energy parametrization. But due to computational expense, for the experiments on real natural language data, we do not use these exact conditionals to perform Gibbs sampling or train exact pseudolikelihood and instead propose MH based samplers described below.

**Relationship with the masked conditionals of MLMs:** Since computing the exact conditional distribution is very expensive, in this section we explore the potential of using MLM conditionals which are easy to compute. Wang and Cho (2019)'s prior attempt to interpret a MLM (like BERT)

as an MRF incorrectly[2] assumes that the positional potentials are independent of each other and hence are not defined on a fully connected clique, i.e. $\phi_t(\mathbf{y}; \theta) = \phi_t(\mathbf{y}_t; \theta)$. This faulty assumption about the factorization of the positional potentials $\phi_t(\mathbf{y}; \theta)$ leads to the following formulation of conditional distribution:

$$p_{mlm}(\mathbf{y}_i|\mathbf{y}_{ni}; \theta) = \frac{\prod_t \phi_t(\mathbf{y}_t; \theta)}{\sum_{w \in \mathbb{V}} \prod_t \phi_t(((\mathbf{y}_{i=w;ni})_t; \theta))} = \frac{\phi_i(\mathbf{y}_i; \theta)}{\sum_{\mathbf{y}_i^0 \in \mathbb{V}} \phi_i(\mathbf{y}_i^0; \theta)} \prod_{t \neq i} \frac{\phi_t(\mathbf{y}_t; \theta)}{\phi_t(\mathbf{y}_t; \theta)} = \text{softmax}(\log \phi_i)$$

This *deficient* conditional distribution for the MRF corresponds to the free conditional distribution $p_{mlm}(\mathbf{y}_t \mid \mathbf{y}_{nt})$ that is obtained by performing a softmax operation over `[MASK]` scores ($\in \mathbb{R}^{\mathbb{V}}$) used in the MLM training objective. These MLM free conditionals do not correspond to the MRF defined by $E_{raw}$ i.e. $p_{mlm}(\mathbf{y}_i \mid \mathbf{y}_{ni}) \neq p(\mathbf{y}_i|\mathbf{y}_{ni}; \theta(E_{raw}))$. In fact, these conditionals need not correspond to *any* consistent MRF over the sequences. As an example, consider a sequence of length 2 with the random variables $\mathbf{y}_1$, $\mathbf{y}_2$ that have a categorical distribution over a vocabulary $\mathbb{V} = \{a; b\}$. The following free conditionals are inconsistent because they do not correspond to any valid joint distribution over $\{\mathbf{y}_1; \mathbf{y}_2\}$: $p(\mathbf{y}_1 \mid \mathbf{y}_2) = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}; p(\mathbf{y}_2 \mid \mathbf{y}_1) = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$. These conditional distributions do not correspond to a valid joint distribution. Intuitively, this is because $\mathbf{y}_2$ is extremely predictive of $\mathbf{y}_1$ but $\mathbf{y}_1$ does not predict $\mathbf{y}_2$ at all from the above conditionals, they cannot characterize a consistent joint distribution over $\{\mathbf{y}_1; \mathbf{y}_2\}$. Furthermore, these conditionals can be shown to violate the Bayes' rule. Using Bayes' rule we observe the following inconsistency:

$$\frac{p(\mathbf{y}_1 = a; \mathbf{y}_2 = a)}{p(\mathbf{y}_1 = a; \mathbf{y}_2 = b)} \propto \frac{p(\mathbf{y}_1 = a \mid \mathbf{y}_2 = a)}{p(\mathbf{y}_1 = a \mid \mathbf{y}_2 = b)} = 99$$

$$\neq$$

$$\frac{p(\mathbf{y}_1 = a; \mathbf{y}_2 = a)}{p(\mathbf{y}_1 = a; \mathbf{y}_2 = b)} \propto \frac{p(\mathbf{y}_2 = a \mid \mathbf{y}_1 = a)}{p(\mathbf{y}_2 = b \mid \mathbf{y}_1 = a)} = 1.$$

[2]This was addressed in their subsequently published erratum: https://bit.ly/2TXS2KM.

It should be noted that prior work on dependency networks (Heckerman et al., 2000) proposed a similar scheme of training the conditionals independently with separate regressions over the latent variables and the inconsistency of such conditionals is well documented (Gelman and Raghunathan, 2001; Dobra et al., 2004; Lowd, 2012).

Wang and Cho (2019) propose that these masked conditionals could be used to define pseudolikelihood ($\prod_{t=1}^{T} p_{mlm}(\mathrm{y}_t \mid \mathrm{y}_{nt};\ )$) maximization objective and subsequently argued that MLM training can be interpreted as stochastic maximization of this *degenerate* pseudolikelihood. However, since these masked conditionals are *deficient* and do not correspond to the *raw-score* energy function $E_{raw}$ as described, MLM training can only be interpreted as stochastic approximation of a *degenerate* pseudolikelihood of the data under $E_{raw}$. Despite the incongruity between MLM training and minimization of $E_{raw}$, we propose to use $E_{raw}$ as one of the parametrizations of the energy function.

### 6.2.2 Effectiveness of MLM objective with $\mathrm{E}_{\mathrm{raw}}$ parametrization on synthetic data

To study the effectiveness of training energy-based models with MLMs and other more faithful objectives, we develop synthetic data whose search space is enumerable. Specifically, we compare the masked language modeling objective with the following two computationally expensive but more accurate approaches using the $E_{raw}$ energy parametrization:

1. Direct maximization of likelihood (**LL**): This approach computes exact loglikelihood of the training data by computing the log partition function under the $E_{raw}$ parametrization via enumeration over the space of all sequences. The size of this space grows exponentially ($\mathcal{O}(|\mathbb{V}|^T)$) with sequence length, $T$. As described in the experiments section, this is computable because of the tractable search space of the synthetic data.

2. Pseudolikelihood maximization (**PLL**): For this approach, we use the exact conditionals,

$p(\mathbf{y}_i|\mathbf{y}_{ni};\ \ )$, that correspond to the $E_{raw}$ energy parametrization as described above. Computing these conditionals is cheaper than computing the exact likelihood but significantly more expensive than computing conditionals in the MLM objective. For the conditional distribution at a position given tokens at all the other positions, $|\mathbb{V}|$ number of passes are run over the decoder compared to just 1 pass for the MLM conditionals. This is computationally feasible because of the fairly small sized vocabulary of the synthetic data.

Our empirical results show that while the masked language modeling objective (MLM) does optimize a decent energy-based model, it still lags behind direct maximization of pseudolikelihood using true conditionals, which further trails direct maximization of the log likelihood i.e. $NLL(MLM) > NLL(PLL) > NLL(LL)$.

In addition to comparing MLM objective with optimizing likelihood and pseudolikelihood, we also propose and compare with another approach described below, that involves training an MLM first and then refining the MLM conditionals for training via exact pseudolikelihood on *smaller amount of in-domain* data.

**Adapting MLM Free Conditionals for Energy Models**

As described above, maximizing the degenerate pseudolikelihood (MLM training) is less computationally expensive than maximizing the true pseudolikelihood and hence MLMs can be learned efficiently on massive amounts of data. Therefore we propose a novel method to project the free conditionals of a pretrained MLM to the true conditionals of the *nearest* well-defined joint distribution. This enables us to effectively benefit from the representational patterns learned by MLMs on large amounts of data for training probabilistic sequence models on smaller potentially domain-specific data. This method focuses on minimizing the KL divergence between the MRF's true condtionals

and the pretrained MLM's conditionals at each position in the training sequences.

$$\min_{\theta} \sum_{t=1}^{T} KL(p_{mlm}(y_t \mid y_{nt}) \| p(y_t \mid y_{nt}; \theta))$$

This approach seems as expensive as maximizing the true pseudolikelihood because we still need to compute the expensive true conditional distributions, $p(y_i|y_{ni}; \theta)$, which require $|mathbbV|$ decoder passes in order to compute the KL divergence. However, because of the MLM pretraining, we need to train with the KL objective with fewer epochs than the epochs needed for maximizing the true pseudolikelihood.

We also experiment with a variant on the above objective which maximizes true PL in addition to minimizing position-wise KL divergence. Using the pretrained MLM in this manner results in fast convergence and well-trained MRF models.

### 6.2.3 Experiments on Synthetic Data

We evaluate the MLM objective's ability to learn well-fitting probabilistic models over sequences against: i) Likelihood maximization (LL) involving exact computation of the normalization constant of the MRF (typically intractable), and ii) Pseudolikelihood (PLL) maximization using exact conditionals (expensive). We also evaluate the effectiveness our proposed approach of adapting MLMs trained on large datasets to train MRFs on smaller datasets.

For fair comparison, all of the systems share the same high-capacity underlying architecture which consists of a bidirectional LSTMs to represent the left and right contexts of the tokens. All the reported results are averaged over 5 runs with different stochastically generated train/test synthetic datasets.

**Synthetic Data**: In order to perform controlled comparisons where the ground truth is known and the likelihood can be exactly computed, we conduct experiments on synthetically generated data (length 5 sequences with vocabulary of size 5) from two different kinds of underlying true

| Eval-metric | Negative Loglikelihood (-LL) | | | | -PLL | -MLM |
|---|---|---|---|---|---|---|
| Train-obj | True | LL | PLL | MLM | PLL | MLM |
| **Synthetic** | | | | | | |
| p-1k | 5.32 | **6.74** | 7.01 | 7.71 | 5.74 | 5.87 |
| p-10k | 5.35 | **6.21** | 6.57 | 7.23 | 4.88 | 5.28 |
| h-1k | 4.97 | **5.07** | 5.16 | 6.01 | 4.58 | 4.46 |
| h-10k | 5.02 | **5.03** | 5.04 | 5.94 | 4.45 | 4.49 |
| h-1k+ | 6.45 | **6.51** | 6.53 | 7.07 | 6.27 | 6.23 |
| h-10k+ | 6.41 | **6.42** | 6.43 | 6.88 | 6.14 | 6.13 |
| **Real** | | | | | | |
| phoneme | - | - | **14.58** | 18.20 | 13.31 | 13.12 |

**Table 6.1:** Comparison of different training objectives on **held-out** datasets. *Eval metrics* are reported for models trained with the specified training objective to maximize the corresponding quantity. Legend: LL-loglikelihood, PLL-Pseudo loglikelihood, and MLM-Degenerate pseudo loglikelihood trained via masked language modeling objective. *True* refers to the ground truth NLL. $p$ refers to the palindrome data generator and $h$ refers to a HMM generator. + refers to the high entropy HMM generator. 1k and 10k refer to the training data sizes.

distributions:

(i) Palindromes: All possible $5^5$ sequences are partitioned into two sets: palindromes and non-palindromes. A Bernoulli draw ($p = 0.8$) chooses whether to sample uniformly from the set of palindromes or a non palindromes.

(ii) HMMs: Data generated from HMMs with 4 hidden states and vocabulary size of 5. We experiment with two different parametrizations of HMMs that correspond to either *low* or *high* entropy marginals.

**Real Data**: We also performed experiments on a *real* dataset–CMU pronunciation dictionary (Lenzo et al., 2007)–to model English phoneme sequences. We removed the stress markers so that the vocabulary size is 39, and only considered sequences of length up to 6 (77653 instances). We made a 90-10 train-test split on this data. The enormous size of the search space precludes training with the exact loglikelihood (both memory and time wise) but it does allow for computation of the exact loglikelihood over the test set for comparison.

**Results**: In Table 6.1, we report the exactly computed negative loglikelihood (NLL) under different models on the synthetic and real datasets with different amounts of data. We compared LL (likeli-

hood maximization)[3], PL (pseudolikelihood maximization with true conditionals), and degenerate pseudolikelihood maximization with free conditionals. The NLL we report for the MLM-trained models is motivated by the success of prior work on reranking sequences (Chen et al., 2017; Salazar et al., 2020), that used the MLM scores to define the potential for an MRF[4] whose distributions could be computed by normalizing over all the possible sequences. While reranking does not require computation of the normalization constant, in our experiments we explicitly enumerated over the search space in order to compare this strategy of using pretrained MLMs as proxies for language models. From the results, we observe that MLMs (Column 5) learn significantly worse probabilistic models than PL or LL based training (columns 3,4). PL maximization performs similarly to LL maximization as the data grows in size. This is despite the fact that both PL and MLM training yield similar PL and MLM values on the test set. It is also interesting to note that as the data size grows, the MLM objective's performance does get closer to the direct pseudolikelihood maximization and the true loglikelihood which indicates that MLMs are not completely incongrunet from the $E_{raw}$ parametrization.

In Figure 6.1, we report the NLL results to test our proposed approach to adapt MLM conditionals to a probabilistic sequence model. For synthetic data, we train an MLM on a large synthetic dataset (100k instances) and for the real data we reserve 65k (out of 75k) instances for training the MLM. Then, we compare the proposed KL projection algorithms to PL-only method for training a probabilistic model on a separate smaller training set (varying size in the Figure). All the methods perform better than just using the pretrained MLM (DPLL (size)). Both the KL projection methods are significantly better than training with PL-only, especially in small data scenarios. This shows that the MLM conditionals could be successfully adapted to probability distributions on small domain-specific data using our proposed approach.

Hence, we demonstrated that the masked language modeling objective training learns poorer

[3]Not tractable on the real data due to large sequence space.
[4]This MRF does not relate to the MLM free conditionals.

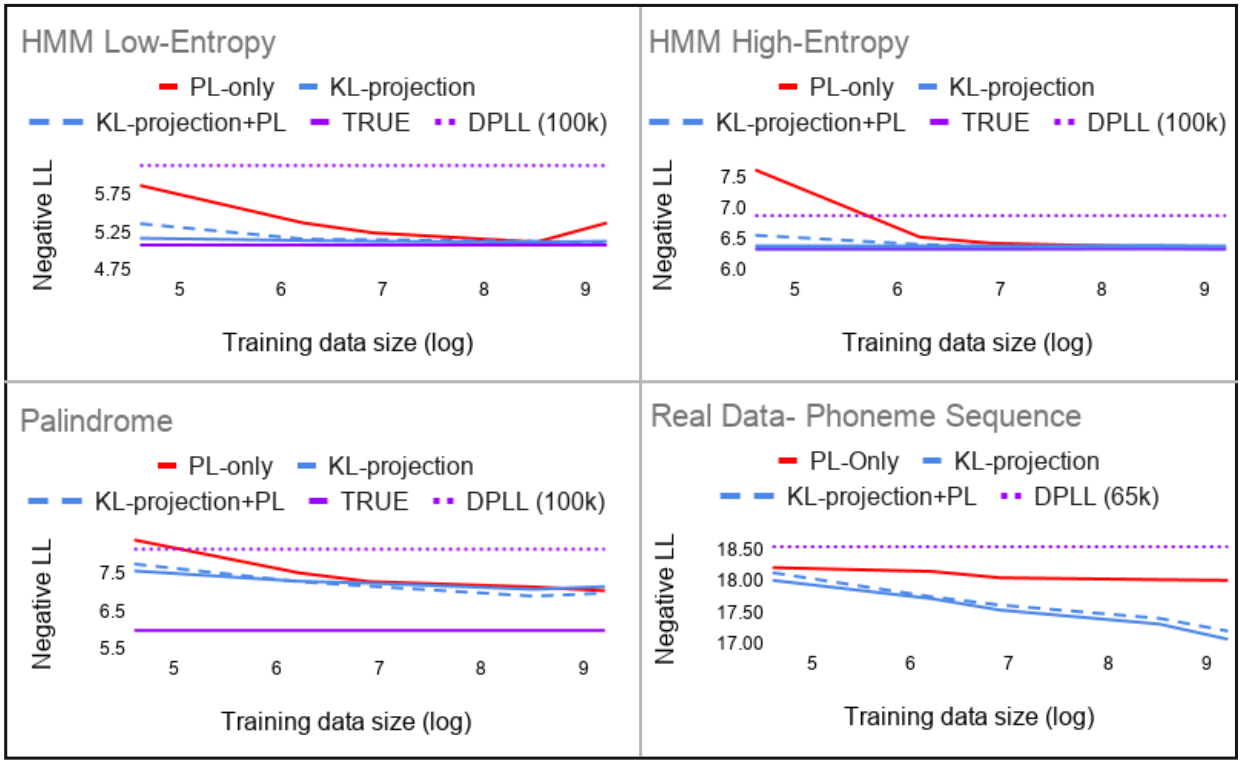


**Figure 6.1:** Comparison of the proposed KL projection method to training with pseudolikelihood (PL-only) from scratch and KL projection combined with PL. TRUE refers to the ground truth NLL and DPLL (# instances) refers to the NLL of pretrained degenerate pseudologlikelihood trained via the MLM objective on the mentioned # of instances. Negative loglikelihood on a held-out set is reported as a function of training data size (in logscale).

energy-based models under $E_{raw}$ than the models learned by minimization of the energy via direct maximization of computationally expensive log lilkelihood or pseudo log likelihood. However, since these MLMs can be trained inexpensively on large amounts of data, the approach to adapt their free conditionals in order to train well-fitting probabilistic models on smaller amounts of data is promising. Also, from these results we can see that the MLM objective, although poorer than direct pseudolikelihood maximization, is still fairly congruent with $E_{raw}$.

### 6.2.4 Locally Normalized Scoring

Having described the $E_{raw}$ parametrization and its link to recently prpoposed energy networks, we turn our attention to another alternative energy parametrization that is related the masked language

models. Recent work (Zhang* et al., 2020) has shown that MLMs like BERT can be used to reliably score a set of sequences. Salazar et al. (2020) and Clark et al. (2020a) developed a scoring scheme to rescore hypotheses proposed by the beam search algorithm and showed downstream improvements over automatic speech recognition (ASR) and machine translation (MT) datasets. The scoring scheme corresponded to masking tokens one-by-one in a left-to-right manner and summing the log-probability of the token at each masked position in the sequence i.e.

$$E_{local}(\mathbf{y};\ ) = -\sum_{t=1}^{|\mathbf{y}|} \log p_{mlm}(\mathbf{y}_i|\mathbf{y}_{ni};\ )$$

This scoring scheme is also implicitly used while performing beam search with the non-autoregressive NMT models proposed in Ghazvininejad et al. (2019), which yielded better BLEU scores when compared to the greedy non-autoregressive baseline. These positive results in prior work suggest that $E_{local}$ is positively correlated with the true sentence scores according to the probabilistic models underlying the trained MLMs.

## 6.3 Background: Metropolis Hastings

In this section we briefly describe the Metropolis Hastings (Hastings, 1970) algorithm. Metropolis Hastings is an MCMC algorithm that provides a recipe for sampling from the distribution $p$ via a proposal distribution $q(\mathbf{y}'; \mathbf{y};\ )$ parametrized by , which defines transition from sequence $\mathbf{y}$ to the sequence $\mathbf{y}'$ in the Markov chain. It assumes the ability to compute the unnormalized score $f(\mathbf{y})$ for every sequence $\mathbf{y}$. At each sampling step we first draw a proposal $\mathbf{y}'$ from the proposal distribution. Then, we either transition to this new state with the acceptance probability $a(\mathbf{y}'; \mathbf{y})$, or repeat the sequence $\mathbf{y}$ in the Markov chain. The acceptance probability is defined as:

$$a(\mathbf{y}'; \mathbf{y}) = \min\left(1; \frac{(\mathbf{y}')\ q(\mathbf{y}; \mathbf{y}')}{(\mathbf{y})\ q(\mathbf{y}'; \mathbf{y})}\right)$$

This sampler satisfies detailed balance and converges to a stationary distribution. Additionally, since it is highly unlikely that the neurally parametrized models like MLMs will assign any sequence a probability 0, it is safe to assume ergodicity of this Markov chain, which guarantees convergence to the desired target energy network distribution $p$.

## 6.4 Masked Conditionals as Proposal Distribution for the MH sampler

As we discuss in Section 6.2.1, the masked conditionals used to train MLMs do not correspond to the two energy formulations we experiment with and are not appropriate for performing Gibbs sampling. In fact, our experiments demonstrate that performing Gibbs sampling using these masked conditionals leads to samples of bad quality, both in terms of BLEU scores and the energies of the generated sequences. However, these conditionals have been shown to be useful for scoring individual sequences and non-autoregressive generation. Therefore, we propose to define the proposal distribution $q(\mathbf{y}'; \mathbf{y})$ for the Metropolis-Hastings sampler by these masked conditionals. More concretely, to transition from the sequence $\mathbf{y}$, we first mask the token in $\mathbf{y}$ at position $i$, i.e., $\mathbf{y}_i = \texttt{[MASK]}$. Next, we do a Transformer decoder pass and get the masked conditionals $p_{mlm}$ at position $i$. Then, the probability of the transition to sequence $\mathbf{y}'$ is the masked probability of the token at the $i$-th position in $\mathbf{y}'$, i.e.:

$$q(\mathbf{y}'; \mathbf{y}) = p_{mlm}(\mathbf{y}'_i | \mathbf{y}_{ni}; \;); \qquad \text{where } \mathbf{y}_{ni} = \mathbf{y}'_{ni} \text{ and } q(\mathbf{y}; \mathbf{y}') = p_{mlm}(\mathbf{y}_i | \mathbf{y}_{ni}; \;);$$

For both Gibbs sampling and MH sampling schemes, we sweep over all the positions while generating sequences of a certain length either sequentially or in a random order. We denote one complete sweep over all the positions in a sequence of length $T$ by the term *epoch*. We summarize our general approach in Alg. 4.

**Algorithm 4** Metropolis Hastings algorithm for MLMs

**Input:** MLM transformer , Energy function $f_E$, MLM conditional proposal $f_{mlm}$ ,sequence length $T$, number of epochs $E$
**Initialize:** $\mathbf{y} \leftarrow$ [MASK]$^T$
$\mathbf{y} \leftarrow$ greedy-decode(MLM($\mathbf{y}$)) . Warm-start with a random sequence
**for** e=0 to E **do**
  **for** t=0 to T **do** . left-to-right or random position selection
    $\mathbf{E}_{\text{old}} \leftarrow f_E(\ (\mathbf{y}))$ . Energy of sequence $\mathbf{y}$, $\mathcal{O}(T)$ op.
    $\mathbf{y}^0 \leftarrow \mathbf{y}, w_o \leftarrow \mathbf{y}_t, \mathbf{y}_t \leftarrow$ [MASK] . Store the t-th token in $\mathbf{y}$ as $w_o$ and mask it.
    $w_n \sim f_{mlm}(\ (\mathbf{y}); t), \mathbf{y}_t^0 \leftarrow w_n$ . Sample $w_n$ from MLM conditional to propose $\mathbf{y}^0$.
    $\mathbf{q}(\mathbf{y}^0; \mathbf{y}) = f_{mlm}(\ (\mathbf{y}); t)[w_n]$, $\mathbf{q}(\mathbf{y}; \mathbf{y}^0) = f_{mlm}(\ (\mathbf{y}); t)[w_o]$
    $\mathbf{E}_{\text{new}} \leftarrow f_E(\ (\mathbf{y}^0))$ . Energy of proposed sequence $\mathbf{y}^0$, $\mathcal{O}(T)$ op.
    $\mathbf{a}(\mathbf{y}^0; \mathbf{y}) \leftarrow \min\ 1; \frac{\mathbf{E}_{\text{new}}\ \mathbf{q}(\mathbf{y}; \mathbf{y}^0)}{\mathbf{E}_{\text{old}}\ \mathbf{q}(\mathbf{y}^0; \mathbf{y})}$ . Acceptance probability of the MC transition.
    **if** $u \sim \mathcal{U}(0; 1), u \leq a$ **then** $\mathbf{y} \leftarrow \mathbf{y}^0$
**Output:** sampled sequence $\mathbf{y}$

**Computational complexity:** Amortizing the encoder cost and the cost of performing a softmax operation, if we denote the cost of doing one Transformer decoder pass over a masked sequence by $C$, then the computational complexity of evaluating MLM conditional is $\mathcal{O}(C)$. For $E$ epochs and a sequence of length $T$, the cost of running a Gibbs sampler is $\mathcal{O}(TEC)$. For the MH sampler, we additionally need to compute the unnormalized scores $(\mathbf{y})$ which, for both the proposed parametrizations of energy, require masking of each position sequentially and running a Transformer decoder pass for each masked sequence. Hence the MH sampler is more computationally expensive with the complexity $\mathcal{O}(T^2EC)$.

### 6.4.1 Variants of Proposal Distribution

We studied our sampler with multiple proposal distributions. While all the variants of proposal distribution rely heavily on the masked conditionals from the pretrained MLM, they have different properties and as shown in the results, they exhibit very different behaviors.

**Varying temperature:** We experiment by changing the entropy of the masked conditionals via a temperature hyperparameter $T$: $q(\mathbf{y}^0; \mathbf{y}; T) = p_{mlm}(\mathbf{y}_i^0 | \mathbf{y}_{ni};\ ; T) = \text{softmax}(\frac{\log\ \ i}{T})$.

**Variation based on Nucleus Sampling:** We experiment with another method of changing the entropy of the masked conditional distribution that is inspired by Nucleus Sampling (Holtzman et al., 2020). It involves defining a nucleus boundary $b$, which prunes out the long tail of the vocabulary that falls outside of the cumulative probability $b$ followed by renormalization over the pruned vocabulary $\mathbb{V}_b$ which is the smallest set such that $\sum_{w \in \mathbb{V}_b} p_{mlm}(\mathbf{y}_i^0 = w | \mathbf{y}_{ni}; \ ) \geq b$.

**Block MH sampling:** Block sampling methods like block Gibbs sampling (Gelfand, 2000) result in better mixing of the Markov chain because they allow for perturbations to multiple variables. While block Gibbs sampling in general is difficult to execute because of the requirement of joint conditional distribution over the block of latent variables given the values of rest of the variables in the system, which is generally intractable to obtain. Our approach, however is flexible to lend nicely to a block sampling scheme that improves mixing and makes the sampling more efficient. In our approach, we mask out multiple tokens in a sequence $\mathbf{y}$ in order to propose a new sequence $\mathbf{y}^0$. Let $\mathcal{I}$ be the set of positions by which $\mathbf{y}$ and $\mathbf{y}^0$ differ. Then, the proposal distribution for the MH sampler is: $q(\mathbf{y}^0; \mathbf{y}) = \prod_{i \in I} p_{mlm}(\mathbf{y}_i^0 | \mathbf{y}_{nI}; \ )$. This proposal distribution is simply a product distrbution of the masked language modelling conditionals at the masked positions which is easy to compute. This makes sampling faster due to parallelization of prediction at several positions, and as shown in our experiments, also results in generation of better samples.

## 6.5 Implementation details

**Pretrained Masked Language Model:** We empirically study the proposed Metropolis Hastings scheme on the conditional generation task of neural machine translation (NMT). For fair comparison we use the pretrained models[5] optimized by a prominent non-autoregressive algorithm–MASK-PREDICT (Ghazvininejad et al., 2019). This algorithm uses a bidirectional Transformer (Vaswani et al., 2017) encoder to encode the source-side sequence and uses the source-side representation to

[5] https://github.com/facebookresearch/Mask-Predict

train the target-side bidirectional transformer-based decoder via the MLM objective. For decoding, conditioned on the source-side encoder representations, it uses a specialized non-probabilistic decoding algorithm which generates text via iterative refinement after randomly masking a subset of tokens at each decoding epoch.

**MCMC details:** For all the sampling baselines, after a burn-in period of 7 epochs, we ran the Markov chain for at least 26 epochs over the dataset. Therefore, for a target sentence of length $T$, we made at least $T \times 33$ proposals in each Markov chain.[6] For all of our sampling results described, we ran at least 5 Markov chains for each configuration described in the subsequent sections and report *averaged statistics* over these runs. In order to run a large number of chains in limited time, for almost all of our experiments, we restrict[7] our translation dataset to source-side sentences of length upto 20. An exception is Table 6.6, in which we report translation performance on full test sets.

**Index selection:** For both Gibbs sampling and the MH samplers we experimented with two methods of selecting the index to mask out: sequential (left-to-right), and random. We observed similar performance, so we describe our experiemtns with random index selection. One key aspect of the random index selection is that for a sequence, we sample without replacement and ensure that all the positions have been sampled once before moving onto the next sequence in the epoch. For block sampling, we randomly sample $b$ positions where $b$ is the block size and similarly keep track of the sampled indices so that they are not sampled multiple times for a sequence in an epoch. We also anneal the block sizes as described in greater detail in section 6.7.3.

**Data:** We performed experiments via translating the validation and test sets of the WMT-14 German-English (De-En), and the WMT-16 Romanian-English (Ro-En) datasets and perform the same tokenization and pre/post-processing as Ghazvininejad et al. (2019).

**Length prediction:** We follow Ghazvininejad et al. (2019), and use a special `[LENGTH]` token

[6]For block-MH sampling variants, we make considerably fewer proposals.

[7]In the limited number of sampling experiments on the full datasets, we observe similar trends as the results on this truncated dataset.

along with the encoder's source side representation to predict the target length. We sample target sentences of different length via batching and padding as necessary.

## 6.6 Metropolis Hastings and *Degenerate* Gibbs Sampling for MLMs

In this section, we empirically compare our proposed Metropolis Sampling approach with both the energy formulations described in Section 6.2 (raw and norm) to the alternative proposed by Wang and Cho (2019) of performing Gibbs sampling with the masked free conditionals. Although, as discussed previously, this Gibbs sampling is incorrect, and we will refer to this method as degenerate Gibbs sampling (deg).

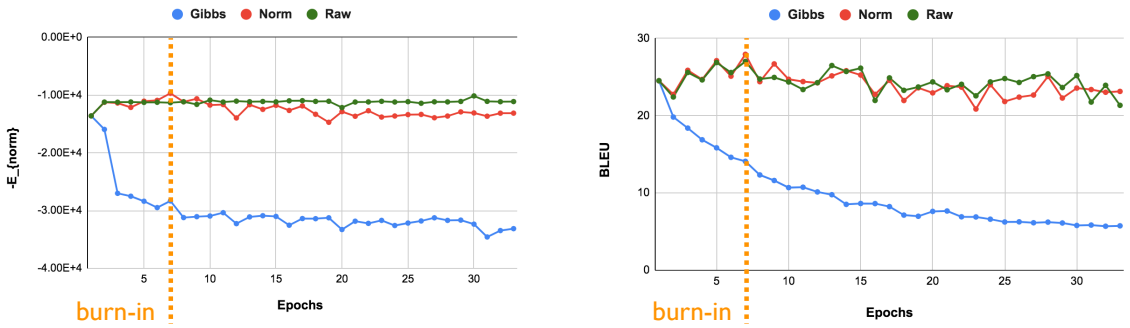


**Figure 6.2:** $-E_{\text{norm}}$ (left) and BLEU scores (right) on De-En (20) as a function of epochs for the two Metropolis Hastings schemes (raw and norm) and the degenerate Gibbs sampling scheme (deg). We compute and report $-E_{\text{norm}}$ even for the samplers with $E_{\text{raw}}$ parametrization.

In Figure 6.2, we notice that although all the samplers start with the same random sequence, the proposed MH samplers generate high quality samples with low energy values and consistently good BLEU scores across all the epochs. The degenerate Gibbs sampler however, keeps on degrading to generating sequences with very low BLEU scores and high energy values[8]. We also observe that

[8]For the results discussed in Figure 6.2, we report standard deviation of the difference between the quantities in the first Markov chain and all the other Markov chains run in our experiments. For our MH based approach, we observed standard deviation of 1.1 in BLEU scores and $0.38 \times 10^3$ for $E_{norm}$.

sequences with high BLEU scores typically have low locally normalized energies[9], which explains the success of prior work in using these locally normalized scores for re-ranking beam search hypotheses and generating sequences with high BLEU scores (Salazar et al., 2020; Ghazvininejad et al., 2019).

Next, we examine the *acceptance ratio* of the MH samplers. We also show the average proportion of *novel MC transition rate*–the ratio of proposals that were distinct from the previous state and accepted–which indicates the entropy of the MCMC transition distribution. The degenerate Gibbs sampler has acceptance probability of $1.0$ by design and a novel transition ratio of $0.36$, which indicates that the MLM conditionals are fairly peaked. Both the MH samplers have high acceptance rates ($0.9$ and $0.91$) but much lower novel transition ratio–$0.11$ for RAW and $0.13$ for NORM. This indicates slow mixing of the MH Markov chain.

## 6.7 Results with Variants of Proposal distributions

### 6.7.1 Effect of temperature

In this section, we explore the effect of temperature on the proposal distributions parametrized by the MLM conditionals as described in Section 6.4.1. We experiment with 5 different temperatures, varying the proposal distributions from high entropy to low entropy. In Table 6.2, we see that the MH sampler performs similarly across all the temperature with the performance improving slightly for lower temperature values, both in terms of the BLEU score and energy values. The degenerate Gibbs sampler exhibits trails behind MH samplers but drastically improves with the lowering temperature values. At low temperatures, it yields decent BLEU scores but it is noteworthy that the energy values are significantly worse than the MH sampler.

Most interestingly, the *novel* transition rates reflect the effect of temperature very clearly. At high temperatures, the degenerate Gibbs sampler never proposes repeating transitions while in

[9]The locally normalized scores ($E_{local}$) and the raw energy scores ($E_{raw}$) are positively correlated.

**Table 6.2:** Average $E_{norm} \times 10^{3}$ energy, novel MC transition rate, and BLEU scores across interleaved epochs for the degenerate Gibbs sampling (deg) and the locally normalized energy MH scheme (Norm) on De-En (20) under MLM proposal distributions with varying temperatures.

| Temp | 2.0 | | 1.5 | | 1.0 | | 0.8 | | 0.5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | norm | deg | norm | deg | norm | deg | norm | deg | norm | deg |
| $E_{norm} \downarrow$ | 12.87 | 32.46 | 10.21 | 29.57 | 11.13 | 31.12 | 9.95 | 21.12 | 7.85 | 17.65 |
| Novel $\leftrightarrow$ | 0.03 | 1.0 | 0.05 | 0.97 | 0.11 | 0.36 | 0.06 | 0.08 | 0.03 | 0.04 |
| BLEU | 25.91 | 14.53 | 24.78 | 10.12 | 24.74 | 9.03 | 25.84 | 24.77 | 27.23 | 26.12 |

stark contrast, the novel transition rate of the MH sampler is extremely low at 0.03. This is because of high rejection rates under the unsuitable high-entropy proposal distribution. While the BLEU/energy results for low-temperature settings seem to suggest that the degenerate Gibbs samplers are practically useful samplers, examining novel transition rates dispels this suggestion. At low temperatures, the novel transition rate is extremely small for the degenerate sampler indicating low-entropy of the MLM based transition distribution which in turn reduces the novel transition rates of the MH sampler as well. Hence, the impressive low-temperature results only corroborate the results of recently proposed greedy non-probabilistic MLM-based generation models like MASK-PREDICT (Ghazvininejad et al., 2019) and do not explore the sequence space in a probabilistic manner.

### 6.7.2 Effect of Nucleus Sampling

Adjusting the nucleus boundary can only decrease the entropy of the MLM proposal distribution. In Table 6.3, we observe effects of low-entropy proposal distribution that are similar to effects of lowering the temperature–decrease in novel transition rate with the samplers fixating around samples with decent BLEU scores and energy values. We also see the similar pattern of the MH samplers being largely robust and yielding good samples under both high and low entropy proposal distributions while the degenerate sampler exhibiting poor performance in high-entropy settings and only improving in low-entropy settings.

**Table 6.3:** Average $E_{norm} \times 10^{-3}$ energy, novel MC transition rate, and BLEU scores energy for the degenerate Gibbs sampling (deg) and the locally normalized energy MH scheme on De-En (20) under MLM proposal distributions with varying nucleus.

| Nucleus | 1.0 | | 0.99 | | 0.95 | | 0.90 | | 0.80 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | norm | deg | norm | deg | norm | deg | norm | deg | norm | deg |
| $E_{norm} \downarrow$ | 11.13 | 31.12 | 10.65 | 30.12 | 10.21 | 28.75 | 9.95 | 18.57 | 9.85 | 18.23 |
| Novel $\leftrightarrow$ | 0.11 | 0.36 | 0.12 | 0.33 | 0.10 | 0.22 | 0.07 | 0.10 | 0.05 | 0.06 |
| BLEU | 24.74 | 9.03 | 24.95 | 14.03 | 26.15 | 18.04 | 27.35 | 23.25 | 27.23 | 23.55 |

These patterns of sensitivity to the proposal distribution's entropy (Tables 6.2,6.3) strongly suggest that while the MLM objective results in conditionals whose mode corresponds to high quality sequences with good BLEU scores and low energy values, these conditionals are poorly calibrated and are not suitable for exploring the distribution over sequences via direct *sampling*. Hence, our proposed sampling technique exhibits robustnness and good performance because it uses the MLM conditionals only to define energy scores and not to directly sample from.

### 6.7.3 Effect of Block MH Sampling

In the results so far, we have observed that while the MH samplers yield good samples, their novel transition rate (**0.11–0.13**) is fairly low which results in slow mixing of the Markov chain. To improve the mixing rate, we experiment with the proposal distribution for block MH sampling as describe in section 6.4.1. Because perturbations in a large number of positions also increase the chance of rejection of the new MH proposal, we balance exploration with tolerable rejection by annealing the number of masked positions with epochs. At the start of the Markov chain, we make large changes, but gradually make smaller changes as the chain progresses. We also, experiment with a *block Gibbs* sampling variant of our degenerate Gibbs sampler. This block Gibbs sampler is incorrect as well, however, it is interesting to study because with temperature $T = 0.0$, it yields the MASK-PREDICT (Ghazvininejad et al., 2019) algorithm. We specify the results while keeping the other settings like temperature and nucleus boundary at their default value of 1.0.

**Table 6.4:** Average BLEU scores, $E_{norm} \times 10^{-3}$, and novel transition rates for the two Block variants of the MH schemes (raw and norm) and the degenerate block Gibbs sampling scheme (deg).

| | Deg | Raw | Norm |
|---|---|---|---|
| $E_{norm} \downarrow$ | 31.18 | 8.08 | 8.17 |
| Novel $\leftrightarrow$ | 0.77 | 0.40 | 0.41 |
| BLEU | 9.03 | 27.12 | 26.78 |

In Table 6.4, we notice that degenerate block Gibbs sampler performs very poorly, while both the MH samplers show drastic improvements in terms of BLEU and the locally normalized scores over previous non-block MH sampling settings under default conditions. Moreover we notice that our block-sampling scheme drastically increases the novel transition rate ($\approx$ **0.12** $\rightarrow$ **0.41**) for our MH samplers. This indicates better exploration of the sequence space.

#### Effect of annealing block size

We linearly anneal the block sizes as a function of iteration i.e. larger block sizes at the beginning for fast mixing and smaller block sizes (eventually reducing to blocksize of 1). The effect of annealing the size of the blocks becomes clearer in Figure 6.3. The initial high acceptance rates indicate fast mixing and iterative refinement of the random initial sequence. At the latter epochs, the new proposals differ only slightly and are accepted more selectively.

## 6.8 Annealing the Energy function: sampling around the modes

In this section, we analyze the effectiveness of our proposed MH samplers by evaluating the samples drawn from regions around the mode of the energy functions. To achieve this, we propose to perform MH sampling from target distributions whose energy values are scaled by low temperatures i.e. $p(\mathbf{y}; \ ; T) \propto e^{\frac{E(\mathbf{y}; \ )}{T}}$. However, such low-entropy target distributions lead to increased rejection rates for the MH samplers. Therefore, we anneal the temperature as a linear function of epochs to gradually decrease the entropy of the target distribution.

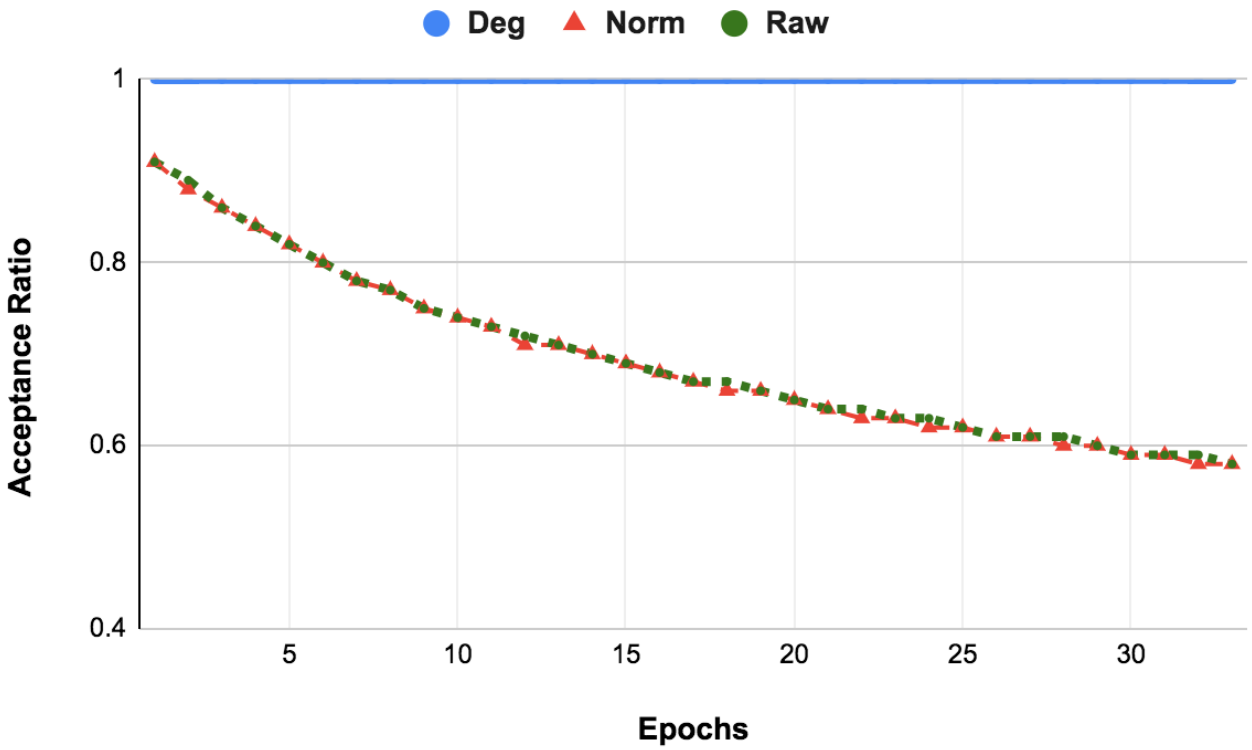


**Figure 6.3:** Acceptance ratio on De-En (20) as a function of epochs for the two Block variants of the Metropolis Hastings schemes (raw and locally normalized energy parametrizations) and the degenerate block Gibbs sampling scheme (deg).

In Figure 6.4 (left, green), we observe that annealing results in dramatic improvements in locally normalized energy scores $E_{norm}$, leading to very low energy values. When comparing the acceptance rates, we see that raw and the locally normalized energy parametrizations behave differently as the target distribution temperature is annealed, with the MH samplers under the raw scores target distribution admitting larger acceptance rates across the epochs. This difference in acceptance rates also manifests itself in the performance in terms of BLEU scores of the samples under two energy parametrizations, with raw energy parametrization yielding higher BLEU scores.

In Table 6.5, we show the effect of different annealing schedules. At each epoch, we subtract either 0:02; 0:04; or 0:06 from the temperature of the target energy function. We see that 0:02 annealing schedule yields the best BLEU scores but interestingly, more aggressive schedules result in better (lower) energy values with lower acceptance rates.

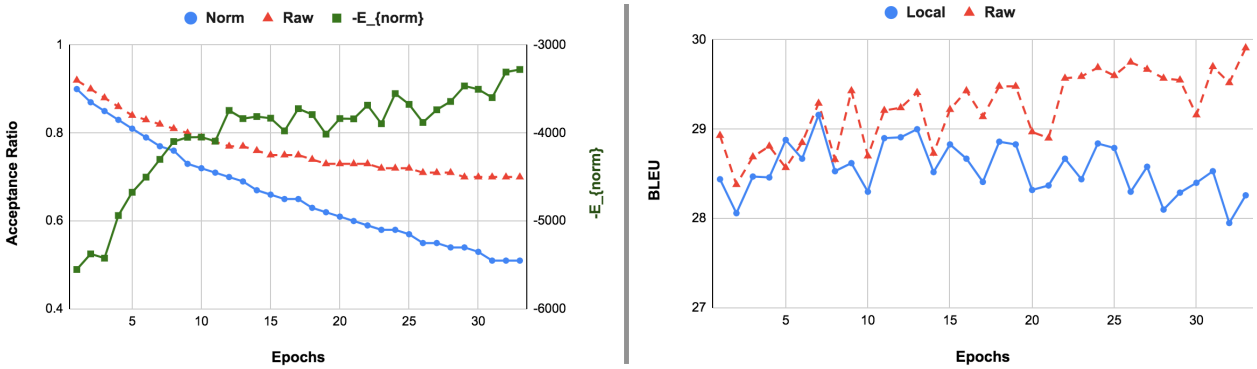


**Figure 6.4:** Comparison as a function of epochs for the two Energy parametrizations (red, blue) for Metropolis Hastings approach with annealing toward modes of target energy functions: $E_{raw}$ (raw) and $E_{local}$ on De-En (20). **Left:** acceptance rates, locally normalized energy (green) as a function of epochs. **Right:** MT performance.

**Table 6.5:** Comparison of different linear annealing strengths via best BLEU and $E_{norm}$, and average acceptance ratio of the two variants of the MH schemes (raw and local) on De-En (20).

| Anneal | | 0.02 | | | 0.04 | | | 0.06 | |
|---|---|---|---|---|---|---|---|---|---|
| | $E_{norm} \downarrow$ | accept | BLEU | $E_{norm} \downarrow$ | accept | BLEU | $E_{norm} \downarrow$ | accept | BLEU |
| Norm | 3277.6 | 0.52 | 29.16 | 3245.65 | 0.45 | 27.58 | 3187.80 | 0.41 | 26.95 |
| Raw | 3146.4 | 0.71 | 29.91 | 3088.65 | 0.62 | 28.32 | 2146.34 | 0.58 | 27.21 |

In Table 6.6, we compare the performance of our annealing-based mode-finding approach on the task of machine translation with other related algorithms. `Warm-start` refers to the greedy replacement of all the mask tokens with the pretrained MLM which is used as the starting sequence for our Markov chains.

### Experimental setup for Table 6.6

Since this experiment focused on obtaining samples from around the mode of the energy parametrizations and comparing them against existing non-autoregressive baselines, to reduce the variance across runs, we used a temperature of proposal distribution as 0.8. This does reduce the diversity in generated samples, but the samples generated have high BLEU scores. For fair comparison, in Table 6.6 we report the performance of degenerate Gibbs sampling with a temperature of 0.8 as well. As can be noticed from the experiments throughout the paper, non-probabilistic greedy

**Table 6.6:** Performance of annealing based approach for sampling around the energy-based distributions' modes. BLEU scores reported on the full De-En and Ro-En test sets.

| | De-En | Ro-En |
|---|---|---|
| Warm-start | 20.27 | 24.38 |
| Degenerate Gibbs (T=0.8) | 27.88 | 29.79 |
| Mask-predict (beam=1, It=10) | 29.27 | 29.95 |
| **Local Energy** | **29.74** | **31.13** |
| **Raw Score Energy** | **30.12** | **30.86** |

mask-based generation approaches tend to perform well when the only desirable attribute of the generated sequence is high BLEU score but fail spectacularly when the MLM conditionals are used for diverse samples characterizing a distribution over the sequences. Another trick to reduce variance was to artificially set the acceptance rate to 1.0 for the first two epochs (after warm-starting) such that all the Markov chains have similar trajectory. We revert back to regular MH sampling acceptance rates after this initialization. As a result, we obtain similar results for Table 6.6 across all the Markov chain runs and the standard deviation in BLEU scores is very low at $\mathbf{0.13}$.

While it performs reasonably well, all the other approaches outperform it. We mainly compare our approach (Local and Raw score Energy) to the MASK-PREDICT algorithm (Ghazvininejad et al., 2019)–a prominent non-probabilistic and non-autoregressive generation technique, which also provides the pretrained conditional MLM for our MH samplers. We outperform both, the degenerate Gibbs sampling with temperature of 0.8 and MASK-PREDICT with beam-size 1, and 10 epochs[10]. Aside from demonstrating the effectiveness of MH sampling approach, the good performance on translation also indicates that our proposed energy parametrizations correspond to good translation models.

[10]Our MH samples are also competitive with the best performance using MASK-PREDICT with beam-size 5, 10 epochs–30.52 De-En, and 31.57 Ro-En.

## 6.9 Conclusion

Our proposed Metropolis-Hastings based sampler enables us to draw high-quality samples from non-probabilistic masked language models. The empirical analysis and success of our approach with the two proposed energy parametrizations on conditional language generation strongly suggests that the optimization of MLM objective results in training of an implicit global energy network that induces probability distribution over the space of sequences and its possible to sample from it using our method. While we primarily focus on sampling and generation, our findings open up avenues for devising more direct, stable and simple training (Deng et al., 2020) procedures for energy-based sequence models inspired from the MLM objectives and our proposed MH sampling scheme.

# Chapter 7

# Closing thoughts

This dissertation describes my work on ameliorating issues like exposure bias and label bias commonly associated with neural autoregressive locally normalized sequence models. While some solutions to directly mitigate exposure bias by making the training of autoregressive models *search-aware* are presented, this dissertation contends that an alternative class of sequence models–globally normalized models–is an appealing class of models with several desirable properties and expressiveness that autoregressive locally normalized models lack.

More specifically, this document describes our work (Goyal et al., 2017, 2018) on search-aware training methods that are amenable to backpropagation for locally normalized neural sequence models that directly ameliorate the issue of exposure bias pertinent to common training methods for autoregressive models like teacher-forcing. The approaches to achieve this at their core, involve continuous relaxation to discontinuous inference procedures like greedy decoding, sampling, and beam search so that these procedures can be incorporated into backpropagation based training procedures which rely on existence of gradients and subgradients. These approaches demonstrate superior performance over teacher-forcing on NLP applications like CCG supertagging, Named Entity Recognition and Machine Translation.

Furthermore, this thesis shifts its focus to an alternative class of globally normalized models that

are unlikely to suffer from exposure bias in the first place and have several advantages over locally normalized models. While this class of models has several desirable properties, it is very difficult to train such models because of the presence of computationally intractable partition function associated with these models that requires enumeration of the whole search space of the possible sequences. In **Chapter 5**, our work (Goyal et al., 2019) on training globally normalized models using a continuous relaxation to beam search for sequences is described which empirically outperform their comparable locally normalized counterparts. We attribute this to label bias experienced by locally normalized models during search-aware training.

Having established the need to explore globally normalized models for discrete sequences with empirical evidence, this document in **Chapter 6** describes our work (Goyal et al., 2021) on more powerful globally normalized models for sequences constructed with the viewpoint of minimization of scalar energy associated with the sequences in the search space of sequences. While training of energy-based sequence models is expensive, brittle and difficult, we focus on interpreting ubiquitous masked language models (MLMs), which have proven to be extremely powerful *encoders* of language, as energy-based models which lets us effectively train powerful globally normalized models via easy-to-optimize masked language modelling objective. This work proposes two energy parametrizations that can be easily computed with pretrained MLMs and introduces an effective Metropolis-Hastings based sampler that uses the MLM free conditionals as Monte Carlo transition distributions for proposals in the MCMC chain to generate high quality samples from the pretrained MLMs. Most importantly, the empirical analysis in this work indicates that masked language modelling objectives are well-suited toward the task of training energy-based globally normalized sequence models and points toward further alternate techniques to train expressive and powerful energy-based models in a tractable manner.

Given that the masked language models are implicitly trained energy-based models, it is a promising avenue to explore the connection between the MLM objective and noise contrastive estimation more deeply. Noise contrastive estimation for sequences would require two major

sequence-level components: 1) The target energy network, and 2) The probability distribution for negative sampling. The masked language objective is a token-level loss and while other approaches like *ELECTRA* (Clark et al., 2020b) and *Electric* (Clark et al., 2020a) perform noise contrastive estimation with a negative sampling distribution inspired by MLM conditional distributions, this view is still restricted purely to estimating distributions over tokens. Another related sequence level model involves training of energy networks to model residual scores which are not accounted for by the pretrained autoregressive components of the full model (Deng et al., 2020). This is done by parametrizing the residuals as bidirectional models with architectures similar to prominent MLMs and using noise contrastive estimation. While this approach is promising, a explicit residual modeling makes the model heavily dependent on the autoregressive component of the models and it is unlikely if the issues associated with the autoregressive models discussed in this document will be addressed with such a training objective. However, it is certainly worthwhile to explore sequence-level noise contrastive estimation with noise distribution parametrized by powerful language encoding models like the masked language models.

Aside from ameliorating exposure bias and label bias issues associated with locally normalized models, energy-based sequence models also enable abandonment of overly restrictive token-level left-to-right generation with autoregressive models. For example, the sampler described in **Chapter 6** generates sets of tokens with much richer conditioning context in arbitrary order. This property is especially appealing for planned and controllable generation of language. Since globally normalized models provide access to whole sequences with regards to generation aspects, this class of models lens itself naturally to non-linear generation which involves seqeunce-level or even paragraph-level constraints like those encountered in story generation (Fan et al., 2018; Yao et al., 2019), poem generation (Ghazvininejad et al., 2016) etc. Specifically, constraints like event-based consistency required in stories, and meter and rhythm based constraints for poem generation are awkward to incorporate into localy normalized models and can only indirectly affect the generation with autoregressive models. However, these constraints are likely to be better encoded easily in a

non-linear and richly contextualized energy-based sequence models, thus providing greater control over the text generation process.

In spite of the virtues of energy-based sequence models, sampling and generating from them is computationally expensive and difficult. As discussed above, while these models can be implicitly trained easily via objectives for masked language modeling or noise contrastive estimation (Gutmann and Hyvärinen, 2010; Ma and Collins, 2018), sampling from them either involves running a Markov chain in an MCMC sampler similar to the one described in **Chapter 6**, or performing importance sampling by training more restricted residual models. Efficient sampling algorithms for discrete symbols are difficult to develop because of absence of any natural manifold and gradients which enable highly efficient methods like Hamiltonian Monte Carlo (Neal et al., 2011) or sampling via Langevin dynamics for continuous random variables. Although approaches like structured prediction energy networks (Belanger and McCallum, 2016) and Langevin dynamics based MCMC sampling (Grathwohl et al., 2021) from discrete MRFs attempt to train networks over discrete symbols by projecting them over a manifold, stable and consistent training of such models that yield good performance over downstream tasks has remained challenging.

Finally, this thesis demonstrates that while an alternative to the dominant autoregressive models–globally normalized models–brings its own sets of limitations, especially those related to computational complexity, this class of models also ameliorates several pathologies associated with autoregressive models and has several desirable properties. Hence, energy-based globally normalized models warrant a strong consideration over the currently default autoregressive models for many applications involving sequence processing and generation.

# Bibliography

Steven Abney, David McAllester, and Fernando Pereira. 1999. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. pages 542–549. 13

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2442–2452. https://doi.org/10.18653/v1/P16-1231. 19, 50, 54, 62

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 244–249. 56

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*. 19

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*. 26, 60

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Interna-*

*tional Conference on Machine Learning*. PMLR, pages 983–992. 67, 94

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1171–1179. 4, 19, 20, 21, 25, 26, 27, 28, 29, 43

Léon Bottou. 1991. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. Ph.D. thesis, Université de Paris XI, Orsay, France. http://leon.bottou.org/papers/bottou-91a. 3, 48

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., volume 33, pages 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. 66

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*. 26, 60

Xie Chen, Anton Ragni, Xunying Liu, and Mark JF Gales. 2017. Investigating bidirectional recurrent neural network language models for speech recognition. In *Proceedings of Interspeech 2017*. International Speech Communication Association (ISCA), pages 269–273. 75

Zhiyi Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics* 25(1):131–160. 13

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar, pages 103–111. https://doi.org/10.3115/v1/W14-4012. 9, 19, 31

Kevin Clark, Minh-Thang Luong, Quoc Le, and Christopher D. Manning. 2020a. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, pages 285–294. https://doi.org/10.18653/v1/2020.emnlp-main.20. 68, 77, 93

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020b. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*. 68, 93

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 111. 54

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning* 75(3):297–325. 19, 25

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*. ACM, pages 169–176. 19

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. 2020. Residual energy-based models for text generation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B1l4SgHKDH. 68, 90, 93

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

*Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 4171–4186. https://doi.org/10.18653/v1/N19-1423. 66, 68

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1370–1380. 56

Adrian Dobra, Chris Hans, Beatrix Jones, Joseph R Nevins, Guang Yao, and Mike West. 2004. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis* 90(1):196–212. 71

Yilun Du and Igor Mordatch. 2019. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689* . 67

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 889–898. https://doi.org/10.18653/v1/P18-1082. 93

Alan E Gelfand. 2000. Gibbs sampling. *Journal of the American statistical Association* 95(452):1300–1304. 80

Alan E Gelfand and Adrian FM Smith. 1990. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association* 85(410):398–409. 66

Andrew Gelman and Trivellore E Raghunathan. 2001. Using conditional distributions for missing-data imputation. *Statistical Science* 15:268–69. 71

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 66, 67, 68, 77, 80, 81, 83, 84, 85, 89

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1183–1191. 93

Matthew R. Gormley, Mark Dredze, and Jason Eisner. 2015. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics (TACL)* . 19

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 366–371. https://doi.org/10.18653/v1/P17-2058. 4, 37, 91

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2019. An empirical investigation of global and local normalization for recurrent neural sequence models using a continuous relaxation to beam search. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 1724–1733. https://doi.org/10.18653/v1/N19-1171. 5, 68, 92

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2021. Exposing the implicit energy networks behind masked language models via metropolis–hastings. *arXiv preprint arXiv:2106.02736* . 6, 65, 92

Kartik Goyal, Graham Neubig, Chris Dyer, and Taylor Berg-Kirkpatrick. 2018. A continuous relaxation of beam search for end-to-end training of neural sequence models. In *Proceedings of AAAI Conference on Artificial Intelligence*. 4, 53, 54, 57, 91

Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. 2021. Oops i took a gradient: Scalable sampling for discrete distributions. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learn-*

*ing*. PMLR, volume 139 of *Proceedings of Machine Learning Research*, pages 3831–3841. http://proceedings.mlr.press/v139/grathwohl21a.html. 68, 94

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B1l8BtlCb. 66, 68

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. pages 297–304. 56, 94

W Keith Hastings. 1970. Monte carlo sampling methods using markov chains and their applications . 67, 77

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. 2000. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research* 1(Oct):49–75. 71

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* . 57

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780. 9, 34

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *LREC*. 42, 58

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rygGQyrFvH. 80

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*. 24, 37

Zhengbao Jiang, Haibo Ding, Jun Araki, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics* https://arxiv.org/abs/2012.00955. 14

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, Vancouver, pages 28–39. https://doi.org/10.18653/v1/W17-3204. 2, 19, 31

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, page 282–289. 12, 50

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data* 1(0). 67

Kevin Lenzo et al. 2007. The cmu pronouncing dictionary. *Carnegie Melon University* 313. 74

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*. pages 592–599. 13

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R Gormley, and Jason Eisner. 2021. limitations of autoregressive models and their alternatives. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 5147–5173. 15

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* . 66

Daniel Lowd. 2012. Closed-form learning of markov networks from dependency networks. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial*

*Intelligence, Catalina Island, CA, USA, August 14-18, 2012*. AUAI Press, pages 533–542. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2313&proceeding_id=28. 71

Zhuang Ma and Michael Collins. 2018. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 3698–3707. https://doi.org/10.18653/v1/D18-1405. 94

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*. 24, 37

Elman Mansimov, Alex Wang, Sean Welleck, and Kyunghyun Cho. 2019. A generalized framework of sequence generation with application to undirected sequence models. *arXiv preprint arXiv:1905.12790* . 68

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '00, page 591–598. 3, 18

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*. 9

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress. http://icml.cc/2012/papers/855.pdf. 56

Radford M Neal et al. 2011. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte*

*carlo* 2(11):2. 94

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619* . 19, 31

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* . 66

Jorge Pérez, Javier Marinković, and Pablo Barceló. 2019. On the turing completeness of modern neural network architectures. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HyGBdo0qFm. 14

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*. 19, 26, 43, 60

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics* 8:842–866. 66

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*. volume 1, page 6. 19, 25

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389. https://doi.org/10.18653/v1/D15-1044. 2, 43

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 2699–2712. https://doi.org/10.18653/v1/2020.acl-main.240. 66, 75, 77, 83

Hava T Siegelmann and Eduardo D Sontag. 1995. On the computational power of neural nets. *Journal of computer and system sciences* 50(1):132–150. 14

Noah A Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics* 33(4):477–491. 13, 53

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, pages 3356–3362. https://doi.org/10.18653/v1/D19-1331. 2

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'14, page 3104–3112. 2, 34, 66

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *Advances in neural information processing systems*. pages 25–32. 38

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147. 26, 42

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research* 6(Sep):1453–1484. 38

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 2819–2826. https://doi.org/10.18653/v1/2020.acl-main.251. 68

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with LSTMs. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 232–237. https://doi.org/10.18653/v1/N16-1027. 42, 58

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, NIPS'17, page 6000–6010. 9, 80

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. *arXiv preprint arXiv:1902.04094* . 66, 68, 69, 71, 82

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 3544–3552. https://doi.org/10.18653/v1/2020.acl-main.326. 2

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 740–745. https://doi.org/10.18653/v1/P18-2117. 14

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1296–1306. https://doi.org/10.18653/v1/D16-1137. 19, 40, 46, 51, 54

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in*

*Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. pages 5754–5764. https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html. 66

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*. volume 33, pages 7378–7385. 93

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTscore: Evaluating text generation with BERT. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SkeHuCVFDr. 66, 77

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *International Conference on Machine Learning*. PMLR, pages 4006–4015. 68

Junbo Zhao, Michael Mathieu, and Yann LeCun. 2017. Energy-based generative adversarial network. In *International Conference on Learning Representations*. 67