Learning to Augment: Non-Autoregressive Translation

Justin Chiu
Cornell Tech
jtc257@cornell.edu

November 29, 2021

Abstract

1 Introduction

High level pitch MLE is bad when the model is misspecified. Outside of language (ie in non-autoregressive image and audio synthesis), GANs have found success at training misspecified models. Within language, GANs are plagued by unstable training (high variance grad estimators). However, seq-KD approaches, which also serve to simplify data, have been successful. We seek to improve upon seq-KD, which does not take into account the model we wish to train. /High level pitch

Learned models of data are often misspecified. When the goal of modeling is not density estimation but some alternative objective, this misspecification may lead to undesirable behaviour under the maximum likelihood objective. In this note, we consider learned data augmentation techniques to edit each data point so that the augmented data is more amenable to a particular model class, while preserving the properties we care about.

A concrete example where maximum likelihood may not be desirable is non-autoregressive translation, which we focus on in this note. In translation, the goal is to translate a source sentence from one language into a sentence in a different target language. The success of translation is measured by the BLEU score between our generated translation and the true reference target sentence. As the BLEU score is not a function of distributional closeness to the true data, translation systems often discard distributional information and output a single translation of a source sentence. However, the training objective is still often maximum likelihood, a function of the model distribution.

In the case of non-autoregressive translation, this results in undesirable behaviour. As non-autoregressive models cannot model output dependencies, resulting models trained via maximum likelihood using the true distribution often have pathological errors due to misspecification, and therefore poor BLEU scores. We aim to resolve this via data augmentation.

Prior work shows that distillation is a successful data augmentation approach. We instead rely on neural editor models [Guu et al., 2017], which are a trainable alternative to data augmentation [Akyürek et al., 2021].

2 Problem Setup

Given a source sentence x, our goal in translation is to model the target sentence y. Our goal is to train a mis-specified non-autoregressive (NAR) model,

$$q(y \mid x) = q(T) \prod_{t=1}^{T} q(y_t \mid x),$$

which, following the knowledge distillation literature, we refer to as the student. As the maximum likelihood estimate (MLE) of the parameters exhibits poor characteristics, in particular poor performance on the primary evaluation metric for translation BLEU, we instead turn to minimizing the reverse KL between the student distribution $q(y \mid x)$ and the true distribution $p^*(y \mid x)$:

$$\begin{aligned} \mathbb{E}_{p^*(x)} \left[D_{\mathrm{KL}} \left[q(y \mid x) || p^*(y \mid x) \right] \right] &= \sum_x p^*(x) \sum_y q(y \mid x) \log \frac{q(y \mid x)}{p^*(y \mid x)} \\ &= \mathbb{E}_{p^*(x)} \left[H(q(y \mid x)) - \sum_y q(y \mid x) \log p^*(y \mid x) \right]. \end{aligned} \tag{1}$$

This objective allows us to ignore difficult-to-model examples from the true distribution, $p^*(y \mid x)$. Unfortunately, the true distribution is unobservable. We instead assume that we have a reasonable approximation of the true distribution, $p(y \mid x)$, which we refer to as the teacher distribution, and optimize the following:

$$\mathbb{E}_{p^*(x)} \left[D_{\mathrm{KL}} \left[q(y \mid x) || p(y \mid x) \right] \right] = \mathbb{E}_{p^*(x)} \left[H[q(y \mid x)] - \sum_y q(y \mid x) \log p(y \mid x) \right]. \tag{2}$$

While both the entropy of the student and cross-entropy between the student and teacher are calculable in theory, the space of translations y is extremely large.

3 Related work

Amortized inference is one method of optimizing such an objective. Recent work has cast the NAR student model as an inference network, and directly trained it to minimize the reverse KL in Equation 2 via stochastic gradient descent [Tu et al., 2020]. Instead of utilizing the high-variance REINFORCE gradient estimator, they instead use biased but lower variance estimators. By performing grid search over different estimators, they found that simple non-stochastic estimators such as a deterministic continuous relaxations via softmax as well as the Straight-Through estimator work best. This particular line of work can be interpreted as primarily searching over the space of student parameters. The search over translations is relaxed into continuous space, allowing for effective gradient estimation given the non-autoregressive student model. Additionally, the student model is initialized with the MLE parameters. The objective is given by

$$\underset{q(y|x)}{\text{minimize}} \sum_x E(x, q(y \mid x)), \tag{3}$$

where $E(x, y) = \log p(y \mid x)$, and the output of $q(x)$ is a $T \times V$ matrix, normalized over columns, giving the probabilities of each word in the translation. The energy function E is relaxed to directly operate on the probabilities, rather than on discrete outputs.

A similar approach is based on sequence-level distillation. Rather than primarily searching over the space of parameters, Sun and Yang [2020] jointly search over the space of autoregressive teacher parameters, non-autoregressive student parameters, as well as translations, by iteratively distilling and searching. Unfortunately, this approach is vastly outperformed by the aforementioned work which avoids searching over teacher parameters and translations, and focuses directly on searching over student parameters.

Other work has attempted to attempted to optimize objectives that do not require full coverage of the true distribution. Work on language generative adversarial networks in particular has attempted to optimize generators and discriminators from scratch using REINFORCE de Masson d'Autume et al. [2019]. The objective optimized in this work, called ScratchGAN, is primarily concerned with marginal language modeling, and is given by

$$\underset{q(x)}{\text{minimize}} \underset{D}{\text{maximize}} \, \mathbb{E}_{p^*(x)} \left[\log D(x)\right] + \mathbb{E}_{q(x)} \left[\log(1 - D(x))\right], \tag{4}$$

where $q(x)$ is the generator and D is the discriminator. This line of work searches over the space of student parameters, discriminators, as well as translations (via sampling, as seen by the expectations).

Sequence level losses.

From these works, we see that the larger the search space, the harder the optimization problem. We aim to build upon the method of Tu et al. [2020], expanding the search space to the space of translations and ablating the continuous relaxation.

4 A Closer Look at Search

The most successful approach to optimizing reverse KL avoided the problem of search over language by using a continuous relaxation and instead focused on search over student parameters [Tu et al., 2020]. We wish emphasize search over language, or formally the summation over translations y in Equation 2.

Our objective is inspired by the reverse KL. Instead of directly optimizing the reverse KL, we introduce auxiliary variables $\hat{Y} = (\hat{y}_i)_i$, which will serve as new translations for each example x_i for the student q to be trained on. Our goal is to find the dataset $\hat{Y}$ by solving

$$\underset{\hat{Y}}{\text{maximize}} \max_{q} \sum_{i} q(\hat{y}_i \mid x_i) \log p(\hat{y}_i \mid x_i), \tag{5}$$

which is the negative reverse KL without an entropy term. Removing the entropy term results in q being allowed to choose a single translation $\hat{y}_i$ for each example. This is important for a more accurate q, as we assumed that q was non-autoregressive and therefore cannot model ambiguity beyond a single word.

We solve this problem by coordinate ascent, iteratively optimizing over $\hat{Y}$ and q. At iteration t, we first find $\hat{Y}^t$ given q^{t-1}. To optimize over $\hat{Y}^t$, we use greedy hill-climbing. At each iteration, we generate $\hat{Y}^t$ by performing search using the teacher model $p(y \mid x)$ and scoring with q^{t-1}. For each example x_i, we search for a set of $J-1$ candidate translations that score higher than the highest scoring translations from $\hat{Y}^{t-1}$, i.e. have high $q^{t-1}(\hat{y} \mid x) \log p(\hat{y} \mid x)$.[1] Given $\hat{Y}^t$, we then find q^t by optimizing

$$\underset{q^t}{\text{maximize}} \sum_i \sum_j q^t(\hat{y}_{i,j} \mid x_i) \log p(\hat{y}_{i,j} \mid x_i). \tag{6}$$

5 Dual of NAR

In this section, we attempt to get rid of explicit length modeling during training using the assumption that we want low entropy solutions. In the primal optimization problem of interest, we wish to maximize

$$L(q) = \sum_y q(y) \log p(y) = \sum_y q(T = |y|) \prod_t q(y_t \mid T) \log p(y) \tag{7}$$

wrt the distribution $q(y) = q(T) \prod_t q(y_t \mid T)$. We can also write q as an exponential family model:

$$q(y) = \frac{\exp(\langle \theta, y \rangle)}{Z} = \frac{\exp([\theta]_T + \sum_t [\theta_{t,T}]_{y_t})}{Z},$$

where $Z = \sum_y \exp(\theta_T + \sum_t \theta_{y_t,t,T})$ is the normalizing constant, and we have $y = (T, y_1, \ldots, y_T)$. The primal problem equivalently optimizes over θ:

$$\begin{aligned} L(\theta) &= \sum_y q(T = |y|) \prod_t q(y_t \mid T) \log p(y) \\ &= \sum_y \frac{\exp([\theta]_T + \sum_t [\theta_{t,T}]_{y_t})}{Z} \log p(y) \\ &= \sum_y \frac{\exp(\langle \theta, \phi(y) \rangle)}{Z} \log p(y), \end{aligned} \tag{8}$$

where $\phi(y) = (T, y_1, \ldots, y_T)$ is a feature function. This form will be helpful, as it makes it clear that $L : \Theta \to \mathbb{R}$ and the structure of $\theta \in \Theta$ is known: $\theta \in \mathbb{N} \times \mathcal{Y}_1 \cdots \times \mathcal{Y}_{T_{\max}}$, where $T_{\max}$ is the max length. The Fenchel-Legendre conjugate of this function is

$$L^*(\mu) = \sup_\theta \langle \theta, \mu \rangle - L(\theta) = \sup_\theta \langle \theta, \mu \rangle - \sum_y \frac{\exp(\langle \theta, \phi(y) \rangle)}{Z} \log p(y). \tag{9}$$

Maximizing over $[\theta]_T$ by setting the derivative to 0, we have

$$[\mu]_T = \sum_y 1(|y| = T) \frac{\exp(\langle \theta, \phi(y) \rangle)}{Z} \log p(y) = \mathbb{E}_q \left[\log p(y) \mid |y| = T \right]$$

[1] Also might have to keep track of high scoring under q and p separately, since q will be changing.

the conditional expected value of $\log p(y)$ given y is length T. Similarly, maximizing over $[\theta_{t,T}]_{y_t}$ yields

$$[\mu_{t,T}]_{y'} = \sum_y 1(y_t = y') \frac{\exp(\langle \theta, \phi(y) \rangle)}{Z} \log p(y) = \mathbb{E}_q \left[\log p(y) \mid y_t = y' \right].$$

Hm, this is not what I wanted. Redo.

deprecated For that subset, perform DFS until J examples that satisfy the likelihood ratio constraint are found (throw in some random samples from past search). Save the stack (memory!). Resume from the given stack when further examples are needed.

This is extremely memory intensive. For each example, we would like to store all translations that satisfy the constraint, and also the stack (i.e. all the next-word probabilities). The stack alone can take around 111 Gb of memory for a small translation dataset (167k sentences, average length of 30, 24k vocab). A common approach to saving memory is the use of amortized Monte Carlo methods.

Things to look out for? While we can prune the search space using the likelihood ratio constraint, there will likely be many examples that satisfy the constraint but are not suitable for the student model q. Additionally, the student model q ties the datapoints $\hat{y}_i$ together globally, meaning that an example may not have been good given the other examples, but changing the other examples may lead to a globally better dataset. Unfortunately, solving this exactly is intractable, so we are stuck with our heuristic.

addled We look to search to solve this problem. However, in practice, the search space is exponentially large and infinite, consisting of all translations y for each source sentence x. However, we can use the log-odds constraint to prune much of the search space. Similar to the admissible heuristic used by Stahlberg and Byrne [2019] to perform exact search, we hope that the constraint prunes enough of the search space to make optimization tractable. Get this empirically: find out how much of the beam /proposals satisfy constraint.

Is there a way to formalize modes?

Exploring modes with editor model

One issue with this is that the score under a model may not be a good indication of an equivalent translation. Possible fix: NLI?

6 Problem Setup

Given a source sentence x, our goal in translation is to model the target sentence y. We approach this problem with the following generative model:

- Given the source sentence x, we choose a proposal translation from a nonautoregressive (NAR) translation model $\hat{y} \sim p(\hat{y} \mid x)$
- We choose a noise vector $z \sim p(z \mid x)$ from a conditional prior
- We choose a final translation $y \sim p(y \mid \hat{y}, z, x)$ from an edit model

Figure 1: The graphical model for the generative model (left) and approximate posterior (right).

This yields the joint distribution

$$p(y, \hat{y}, z \mid x) = p(y \mid \hat{y}, z, x)p(\hat{(y)} \mid x)p(z \mid x)$$

The generative model is shown in Figure 1.

For training, optimizing the log marginal likelihood of the translation

$$p(y \mid x) = \sum_{\hat{y}} \int_z p(y, \hat{y}, z \mid x),$$

is intractable due to the combinatorial space of possible translations and the integral over z. We instead resort to approximate inference, in particular variational inference. We introduce the approximate posteriors $q(z \mid y, x)$ and $q(\hat{y} \mid z, y, x)$, yielding the following lower bound

$$\begin{aligned}
\log p(y \mid x) &= \log \sum_{\hat{y}} \int_z p(y, \hat{y}, z \mid x) \\
&= \log \sum_{\hat{y}} \int_z q(\hat{y} \mid z, y, x) q(z \mid x) \frac{p(y, \hat{y}, z \mid x)}{q(\hat{y} \mid z, y, x) q(z \mid x)} \\
&\geq \sum_{\hat{y}} \int_z q(\hat{y} \mid z, y, x) q(z \mid x) \log \frac{p(y, \hat{y}, z \mid x)}{q(\hat{y} \mid z, y, x) q(z \mid x)} \\
&= \mathbb{E}_{q(\hat{y} \mid z, y, x) q(z \mid x)} \left[\log \frac{p(y, \hat{y}, z \mid x)}{q(\hat{y} \mid z, y, x) q(z \mid x)} \right] \\
&= \mathbb{E}_{q(\hat{y} \mid z, y, x) q(z \mid x)} \left[\log p(y \mid \hat{y}, z \mid x) \right] - D_{\mathrm{KL}} \left[q(\hat{y} \mid z, y, x) || p(\hat{y} \mid x) \right] - D_{\mathrm{KL}} \left[q(z \mid y, x) || p(z \mid x) \right].
\end{aligned}$$

Approximate edit vector posterior We parameterize $q(z \mid y, x)$ with a Transformer and some continuous distribution, and estimate its gradient with a pathwise derivative estimator. Alternatively we can use a discrete distribution and REINFORCE (enumeration).

Approximate translation proposal posterior For $q(\hat{y} \mid z, y, x)$, we have a couple options for parameterization. Before we discuss those options, we note non-autoregressive models (both $q(\hat{y} \mid z, y, x)$ and $p(\hat{y} \mid x)$) require first predicting length, then each word independently, resulting in the following factorization: $p(\hat{y}) = p(l) \prod_t (y_t \mid l)$. We can now cover parameterizations and gradient estimators. There are two axes of variation: do we perform parameter sharing, and do we relax estimators. For parameter sharing, that means we can

- Introduce new parameters for $q(\hat{y} \mid l, z, y, x)$
- Use the prior $q(\hat{y} \mid l, z, y, x) = p(\hat{y} \mid l, x)$

Additionally, we can use either the REINFORCE gradient estimator or use Concrete relaxations of each $p(\hat{y}_t \mid l)$. The latter would allow us to estimate gradients with a pathwise derivative estimator. The former would require some care due to high variance. One approach to variance reduction could be to use a baseline (self-critical, LOO, RELAX), in combination with multi-sample estiamtors (Rao-Blackwellization or sampling without replacement or stochastic beam search).

Fixing a fatal Flaw: constraining $\hat{y}$ The augmented data $\hat{y}$ is completely unconstrained in this scenario, leading to the possibility that the model can learn an informative $\hat{y}$ that is not useful for translation. Normally, this would be prevented by the likelihood $p(y \mid \hat{y}, z)$. However, if the likelihood is too flexible (as will likely be the case for us), $\hat{y}$ must be constrained via other means. In particular, we can use posterior regularization. This could entail adding an expectation constraint that ensures that the approximate translation proposal posterior does not deviate too far from a pretrained translation model $p^*(y \mid x)$ trained via normal means:

$$\mathbb{E}_{q(\hat{y}|z,y,x)}\left[\log \frac{q(\hat{y} \mid z, y, x)}{p^*(\hat{y} \mid x)} \leq \delta\right].$$

Alternatively, we can fix a pretrained NAT prior $p(\hat{y} \mid x)$ and train the edit model to completion, then train another model on the generated data.

7 Experiments

Toy experiments

- Take set of strings and tie-break synonyms
- Augment PCFG-generated data into NAT (tree synonyms, movement)

Improving distillation Start with NAT model trained with distillation, and output of distillation's teacher model as $\hat{y}$. Run variational EM and see if anything improves.

Improving EM+Distillation Start with NAT model trained with EM+distillation, and output of distillation's teacher model as $\hat{y}$. Run variational EM and see if anything improves.

Denoising Data2Text Data2Text datasets often contain noisy text that is not represented in the data. Can we learn to filter that out? How would we measure success?

8 Optimizing BLEU

Formulate the following optimization problem

$$\begin{aligned} &\text{maximize } \max_{q} \sum_{i} \log q(\hat{y}_i \mid x) \\ &\text{subject to BLEU}(y, \hat{y}) \geq \delta. \end{aligned} \tag{10}$$

Unfortunately, neither the objective nor the constraint are separable. The objective contains a nested optimization problem, so decoupling that will not be fruitful. However, we can decouple an approximation of the constraint.

9 Problem Setup: Bilevel Optimization (not updated)

Source sentence x, target sentence y. True translation distribution $p(y \mid x)$.

We propose to, given a family of student models $q_\theta(y \mid x)$, learn an edit model $q_\phi(\hat{y} \mid y, x)$ whose conditional distribution over $\hat{y}$ is easier for the student model q_θ to learn. Learning an intermediate distribution will allow the student model to focus on only modeling the important aspects of the true distribution.

To accomplish this, we propose to solve the following optimization problem

$$\operatorname*{argmin}_{\phi} D_{\mathrm{KL}} \left[p(y \mid x) || \sum_{y} q_\phi(\hat{y} \mid y, x) p(y \mid x) \right] + \min_{\theta} D_{\mathrm{KL}} \left[q_\phi(\hat{y} \mid y, x) || q_\theta(\hat{y} \mid x) \right]. \tag{11}$$

This is a bilevel optimization problem, where we want to find the edit model that is able to balance faithfulness to the true data (the first term) as well as learnability of the student model (the second term). We refer to Equation 11 as the outer problem, and the second term as the inner problem:

$$\operatorname*{argmin}_{\theta} D_{\mathrm{KL}} \left[q_\phi(\hat{y} \mid y, x) || q_\theta(\hat{y} \mid x) \right]. \tag{12}$$

The first term of the outer problem, the KL divergence between the true distribution and the marginal posterior of the edit distribution, serves as a regularizer and encourages the marginal distribution of the edit distribution combined with the data distribution to be close to the original data distribution. As this setting may be both too restrictive as well as not example-specific, we also consider the following variation:

$$\operatorname*{argmin}_{\phi} \sum_{y} \Omega(y, q_\phi(\hat{y} \mid y, x)) + \min_{\theta} D_{\mathrm{KL}} \left[q_\phi(\hat{y} \mid y, x) || q_\theta(\hat{y} \mid x) \right], \tag{13}$$

allowing us to perform instance-level regularization rather than distributional.

10 Method 1: The Exact Setting

We first make the simplifying assumption that we can solve the inner problem exactly. In order to then solve the full outer problem, we will differentiate through the solution of the inner problem. In full generality, this would require an application of the implicit function theorem, or approximations through sampling or incomplete optimization.

10.1 Experiment: Length-2 Binary Strings + Non-autoregressive Student

Our first experiment will determine whether training an edit model is possible in a very simple setting, where we can compute the solution to the inner problem (training the student model given an edit model) in closed form, and subsequently differentiate through that procedure.

We restrict our attention to distributions over target sentences that are binary strings of length 2 without conditioning on a source sentence. In particular, our true distribution takes the form

$$y = b_1 \cdot 00 + b_2 \cdot 01 + b_3 \cdot 10 + b_4 \cdot 11,$$

where $b \sim \mathrm{Cat}(\lambda)$, represented as a one-hot sample (meaning only one of b_i can be 1, while the rest must be 0). As we are interested in disambiguation, we first focus on the case where $b_1, b_4 \approx 0$, and $b_2 < b_3$. The edit model then takes the form

$$q_\phi(\hat{y} \mid y) = [\phi]_{\hat{y},y},$$

where $\hat{y}, y \in \{0,1\}^2 = \mathcal{Y}$ and $\phi \in [0,1]^{\mathcal{Y}\times\mathcal{Y}}$. The student is nonautoregressive, and has the form

$$q_\theta(\hat{y}) = \prod_t q_\theta(\hat{y}_t) = [\theta]_{1,\hat{y}_1} \cdot [\theta]_{2,\hat{y}_2},$$

where $\theta \in [0,1]^{2\times\mathcal{Y}}$.

In this setting, we can exactly compute the solution of the inner problem,

$$\operatorname*{argmin}_\theta \mathbb{E}_y \left[D_{\mathrm{KL}} \left[q_\phi(\hat{y} \mid y) \right] || q_\theta(\hat{y}) \right] = \operatorname*{argmin}_\theta \mathbb{E}_y \left[\sum_{\hat{y}} q_\phi(\hat{y} \mid y) \left(\log q_\phi(\hat{y} \mid y) - \log q_\theta(\hat{y}) \right) \right],$$

and differentiate through it. The minimizer is given by

$$\begin{aligned}
[\theta]_{1,0} &= \sum_y \sum_{\hat{y}} p(y) q_\phi(\hat{y} \mid y) 1(\hat{y}_1 = 0) \\
[\theta]_{2,0} &= \sum_y \sum_{\hat{y}} p(y) q_\phi(\hat{y} \mid y) 1(\hat{y}_2 = 0) \\
[\theta]_{t,1} &= 1 - [\theta]_{t,0},
\end{aligned}$$

which is differentiable wrt ϕ.

Results We find that the editor model is able to disambiguate the true data and produces a new data distribution with no dependencies across time, resulting in a distribution that is easier for the student to match. We see the results of training the editor model with the following conditions:

- $b = [0.001, 0.45, 0.54, 0.001]$
- Editor is initialized to uniform everywhere except the transitions from [0,1] and [1,0], where transitions to [0,0] and [1,1] are given low mass in order to speed up training
- We recompute the parameters of the student by solving the inner problem at every iteration
- We solve the outer problem via gradient descent (1k iterations, learning rate of 1e-3)

We see in Figure 2 (right) that the editor learns to bias the data further towards $[1, 0]$. This results in the student model being very certain, as seen in Figure 3 (right). In this very simple setting, the bilevel optimization formulation successfully increases the assymmetry of the data.

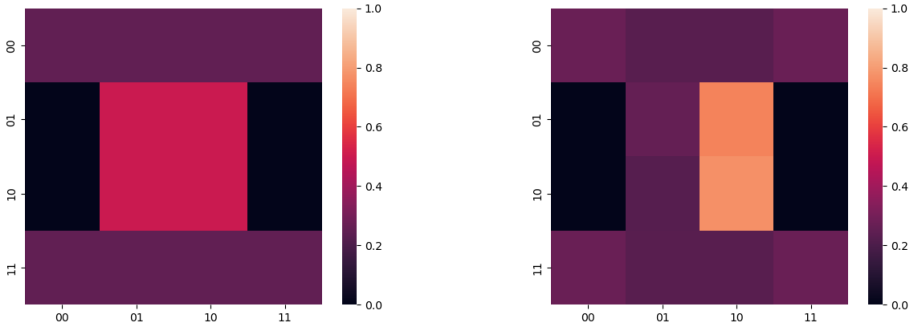

Figure 2: The emission probabilities for the editor model at the start of training (left) and end of training (right).

However, in settings where the true data distribution is less skewed, the objective does not allow the edit model to deviate too far. Figures 4 and 5 show the results when $b = [1e-3, 0.5, 0.5, 1-e3]$. In this setting, both the data distribution and edit distribution are symmetric, resulting in the student remaining at a standstill. This is because any movement away from the true distribution would incur too large a penalty in the first term of the objective, $D_{\text{KL}}\left[p(y)||\sum_y q(\hat{y} \mid y)p(y)\right]$.

References

E. Akyürek, A. F. Akyürek, and J. Andreas. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PS3IMnScugk`.

C. de Masson d'Autume, M. Rosca, J. W. Rae, and S. Mohamed. Training language gans from scratch. *CoRR*, abs/1905.09922, 2019. URL `http://arxiv.org/abs/1905.09922`.

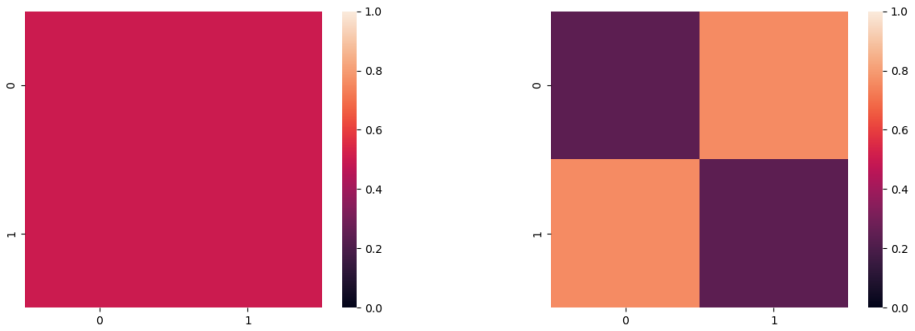

Figure 3: The emission probabilities for the student model at the start of training (left) and end of training (right).

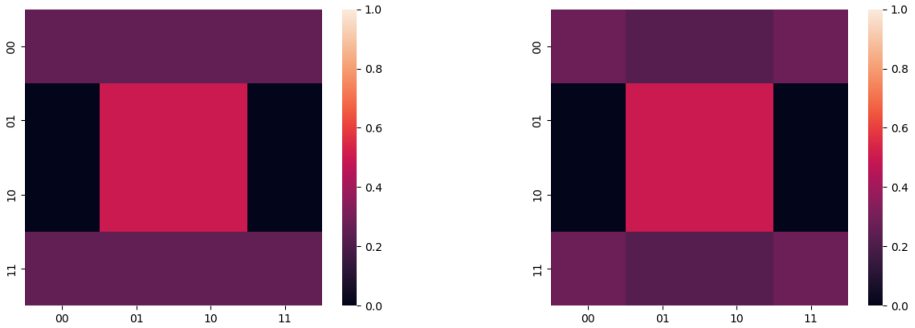

Figure 4: The emission probabilities for the editor model at the start of training (left) and end of training (right) with an even true data distribution.

K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. Generating sentences by editing prototypes. *CoRR*, abs/1709.08878, 2017. URL `http://arxiv.org/abs/1709.08878`.

F. Stahlberg and B. Byrne. On NMT search errors and model errors: Cat got your tongue? *CoRR*, abs/1908.10090, 2019. URL `http://arxiv.org/abs/1908.10090`.

Z. Sun and Y. Yang. An EM approach to non-autoregressive conditional sequence generation. *CoRR*, abs/2006.16378, 2020. URL `https://arxiv.org/abs/2006.16378`.

L. Tu, R. Y. Pang, S. Wiseman, and K. Gimpel. ENGINE: energy-based inference networks for non-autoregressive machine translation. *CoRR*, abs/2005.00850, 2020. URL `https://arxiv.org/abs/2005.00850`.

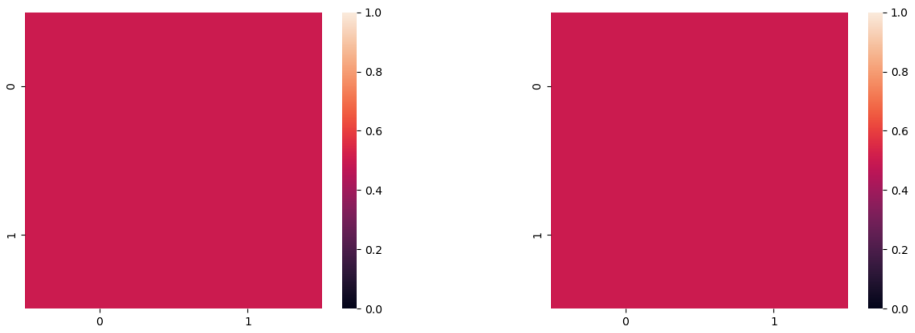

Figure 5: The emission probabilities for the student model at the start of training (left) and end of training (right) with an even true data distribution.