Learning to Augment: Non-Autoregressive Translation

Justin Chiu
Cornell Tech
jtc257@cornell.edu

November 12, 2021

Abstract

Abstract

1 Introduction

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 of a setup 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. We approach this problem with the following generative model:

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

- 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

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

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

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:

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

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.

3 Experiments

Toy experiment Augment PCFG-generated data into NAT.

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.

Experiments from scratch Run variational EM and see if anything happens from scratch.

4 Problem Setup: Search (Only)

Fix the teacher model, frame the problem following Sun and Yang [2020]: Find the dataset $\hat{Y} = (\hat{y}_i)_i$ by solving

$$\begin{aligned} &\text{minimize} \sum_i \log q(\hat{y}_i \mid x) \\ &\text{subject to} \sum_i \log \frac{p(y_i \mid x)}{p(\hat{y}_i \mid x)} \geq \epsilon, \end{aligned} \tag{1}$$

which looks for a dataset that is high probability under a non-autoregressive student model $q(y \mid x)$, but also high probability under a baseline autoregressive model $p(y \mid x)$ (relative to the ground truth translation).

5 Problem Setup: Bilevel Optimization (deprecated)

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

$$\operatorname*{argmin}_\phi D_{\mathrm{KL}} \left[p(y \mid x) || \sum_y q_\phi(\hat{y} \mid y, x) p(y \mid x) \right] + \min_\theta D_{\mathrm{KL}} \left[q_\phi(\hat{y} \mid y, x) || q_\theta(\hat{y} \mid x) \right]. \tag{2}$$

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 2 as the outer problem, and the second term as the inner problem:

$$\operatorname*{argmin}_\theta D_{\mathrm{KL}} \left[q_\phi(\hat{y} \mid y, x) || q_\theta(\hat{y} \mid x) \right]. \tag{3}$$

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:

$$\operatorname*{argmin}_\phi \sum_y \Omega(y, q_\phi(\hat{y} \mid y, x)) + \min_\theta D_{\mathrm{KL}} \left[q_\phi(\hat{y} \mid y, x) || q_\theta(\hat{y} \mid x) \right], \tag{4}$$

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

6 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.

6.1 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.

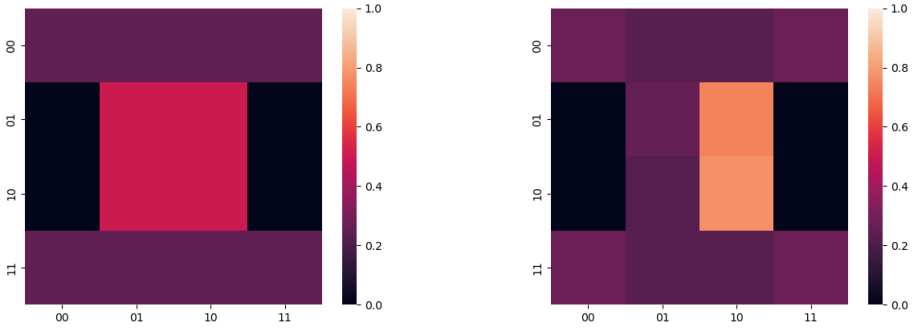

Figure 2: The emission probabilities for the editor model at the start of training (left) and end of training (right).

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.

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.

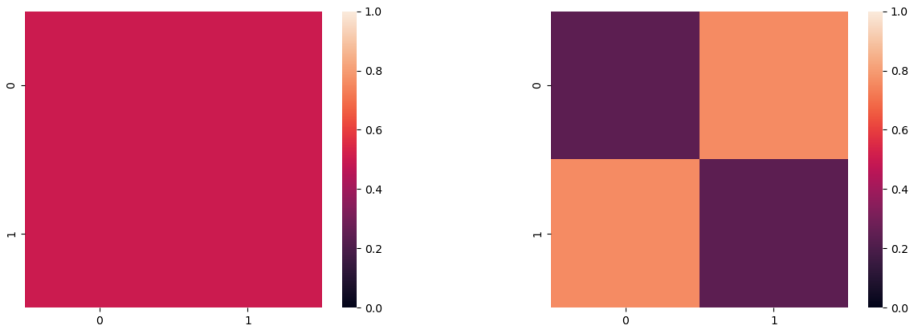

Figure 3: The emission probabilities for the student model at the start of training (left) and end of training (right).

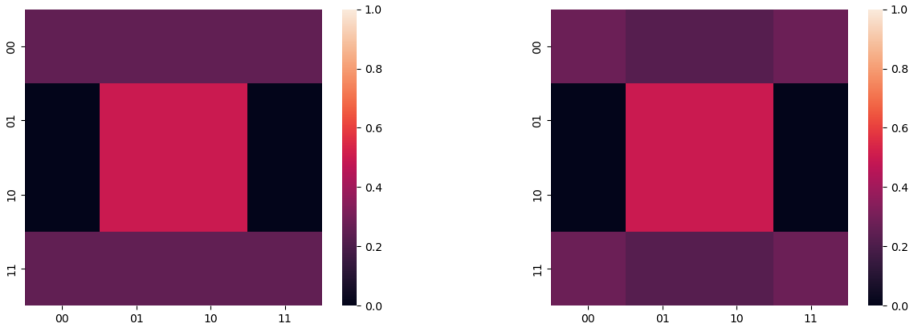

Figure 4: The emission probabilities for the editor model at the start of training (left) and end of training (right) with an even true data distribution.

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.

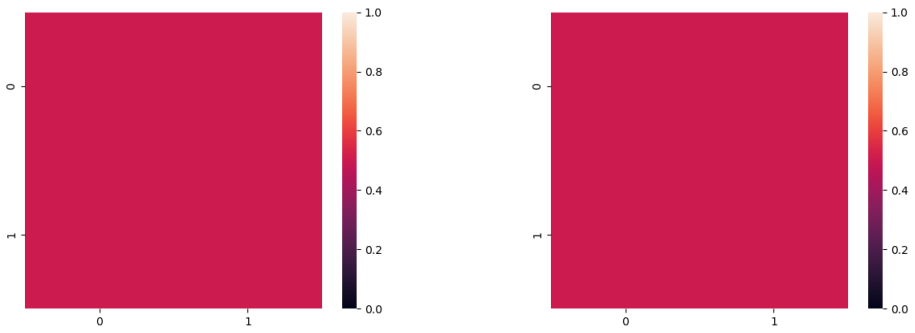

Figure 5: The emission probabilities for the student model at the start of training (left) and end of training (right) with an even true data distribution.