Learning to Augment: Non-Autoregressive Translation

Justin Chiu
Cornell Tech
jtc257@cornell.edu

November 5, 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 hopefully 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

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

$$\operatorname*{argmin}_{\phi} D_{\mathrm{KL}}\left[p(y \mid x) || q_\phi(\hat{y} \mid y, x)\right] + \min_{\theta} D_{\mathrm{KL}}\left[q_\phi(\hat{y} \mid y, x) || q_\theta(\hat{y} \mid x)\right]. \tag{1}$$

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

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

3 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.

3.1 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:

- $b = [0.001, 0.4, 0.59, 0.001]$
- 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 1 (right) that the editor learns to remove instances of 01 from the data in favor of consistently outputting 10. This results in the student model being very certain, as seen in Figure 2 (right). We conclude that, in this very simple setting, the bilevel optimization formulation successfully disambiguates the data.

Decoding: work back into problem setup later

Given data consisting of (x, y) pairs, our goal is to learn a model $q_\theta(y \mid x)$ such that it maximizes the functional

$$F(q) = \mathbb{E}_{p(x,y)} \left[\operatorname*{argmax}_{\hat{y}} d(q_\theta(\hat{y} \mid x), y) \right], \tag{3}$$

where $p(x, y)$ is the data distribution and d is some measure of correctness between our prediction $\hat{y}$ and the true output y.

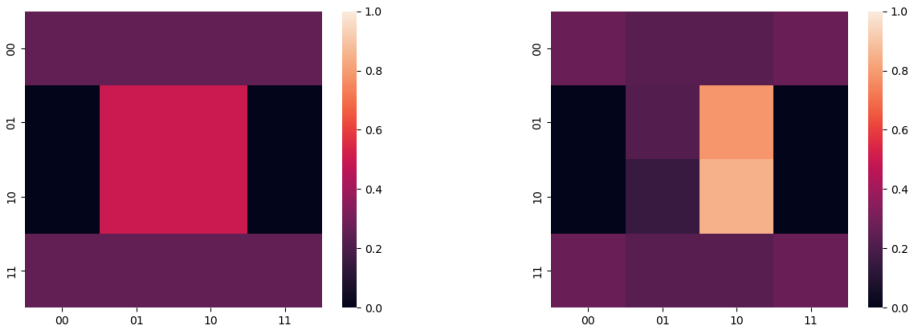

Figure 1: The emission probabilities for the editor model at the start of training (left) and end of training (right).

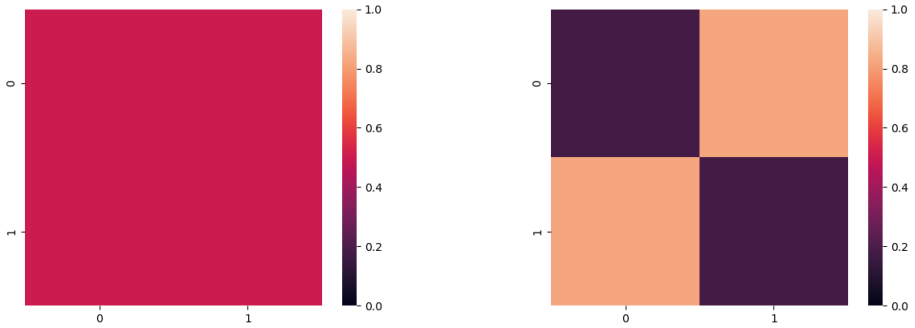

Figure 2: The emission probabilities for the student model at the start of training (left) and end of training (right).

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.