SEQUENCEMATCH: IMITATION LEARNING FOR AUTOREGRESSIVE SEQUENCE MODELLING WITH BACKTRACKING

Chris Cundy[1] **Stefano Ermon**[1]
[1]Department of Computer Science, Stanford University
`{cundy, ermon}@cs.stanford.edu`

ABSTRACT

In many domains, autoregressive models can attain high likelihood on the task of predicting the next observation. However, this maximum-likelihood (MLE) objective does not necessarily match a downstream use-case of autoregressively generating high-quality sequences. The MLE objective weights sequences proportionally to their frequency under the data distribution, with no guidance for the model's behaviour out of distribution (OOD): leading to compounding error during autoregressive generation. In order to address this compounding error problem, we formulate sequence generation as an imitation learning (IL) problem. This allows us to minimize a variety of divergences between the distribution of sequences generated by an autoregressive model and sequences from a dataset, including divergences with weight on OOD generated sequences. The IL framework also allows us to incorporate backtracking by introducing a `backspace` action into the generation process. This further mitigates the compounding error problem by allowing the model to revert a sampled token if it takes the sequence OOD. Our resulting method, SequenceMatch, can be implemented without adversarial training or architectural changes. We identify the SequenceMatch-χ^2 divergence as a more suitable training objective for autoregressive models which are used for generation. We show that empirically, SequenceMatch training leads to improvements over MLE on text generation with language models and arithmetic.

1 INTRODUCTION

Autoregressive models such as the GPT series of causally masked transformers (Brown et al., 2020; Radford et al., 2018) are able to perform a variety of downstream tasks such as question-answering, translation, and summarization, after simply training on a large corpus of text with the objective of predicting the next token given the previous tokens. However, autoregressive language models suffer from a variety of pathological behavior when deployed on the task of free-form text generation (Holtzman et al., 2019; Welleck et al., 2019), particularly at lower generation temperatures or with smaller models. These include generating the same token or series of token repeatedly, or generating gibberish outputs. This problem of neural text degeneration has been linked to the training objective for LLMs, which trains a conditional distribution for the next token given a (partial) sentence (Fu et al., 2021). When deployed in an autoregressive fashion, the model has its own outputs as inputs, resulting in a compounding error problem that rapidly takes the model out of distribution (OOD). This compounding error problem is also a key issue in the imitation learning subfield of reinforcement learning, where the goal is to learn a policy (a distribution over next actions given the past) which results in trajectories similar to a set of expert trajectories. The approach of directly matching the expert's actions leads to a compounding error (Ross et al., 2011), which has led to several works proposing to address this problem by minimizing alternative divergences (Arora et al., 2022; Shi et al., 2018). These alternative divergences encourage the policy to return to expert states if the generated trajectory starts to diverge from them. We argue that two key issues can prevent autoregressive models trained with maximum-likelihood from generating fluent sequences at evaluation time. First is the divergence measure used to evaluate the difference between the model and the data distribution. Because the MLE loss does not have any contribution from OOD sequences, the behavior of

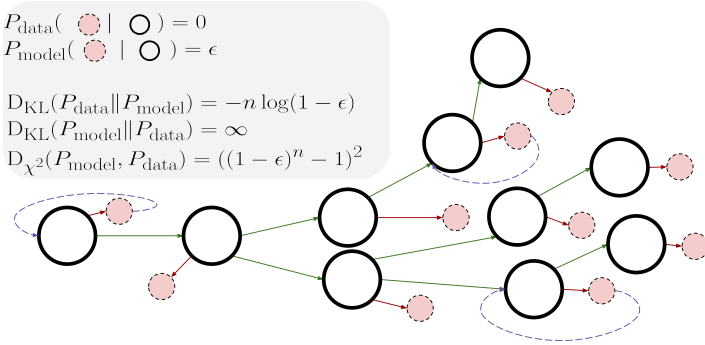
$P_{\text{data}}(\,◌ \mid ○\,) = 0$
$P_{\text{model}}(\,◌ \mid ○\,) = \epsilon$
$\mathrm{D}_{\mathrm{KL}}(P_{\text{data}} \| P_{\text{model}}) = -n \log(1-\epsilon)$
$\mathrm{D}_{\mathrm{KL}}(P_{\text{model}} \| P_{\text{data}}) = \infty$
$\mathrm{D}_{\chi^2}(P_{\text{model}}, P_{\text{data}}) = ((1-\epsilon)^n - 1)^2$

Figure 1: A toy model of an autoregressive generation problem, such as language modelling. Our task is to learn a set of conditional distributions that continue the sequence similarly to those sequences in the dataset (green arrows), and avoid incorrect next tokens (red arrows). Our method trains against divergences that more heavily punish out-of-distribution sequences. We additionally introduce a backspace action which can backtrack from an erroneous token (dashed purple arrows).

the model on OOD sequences is unconstrained. We address this by minimizing the χ^2-divergence between a mixture of the data and autoregressively generated sequences. This divergence is known to perform much better than MLE in imitation learning (Garg et al., 2021; Al-Hafez et al., 2023).

Secondly, if a model generates an OOD token, there may be no natural continuation which is similar to sequences from the data distribution, and so the model may be unable to return to the data distribution even if our χ^2-divergence encourages this. To address this, we augment the generation process with a backspace action `<bkspc>`, which deletes the previous token, and allows the model to correct for erroneous generations. By incorporating recent work in non-adversarial imitation learning (Garg et al., 2021), our method, *SequenceMatch*, is able to train autoregressive models against alternative divergences such as the χ^2-mixture divergence while augmenting the policy with a backspace action. The SequenceMatch loss is a fully supervised loss without adversarial training, and can be applied on top of pretrained models as a finetuning step. To summarize our contributions:

- We formulate the sequence generation problem as an imitation learning (IL) problem, and formulate a general non-adverarial objective for minimizing divergences between occupancy measures based on (Garg et al., 2021), handling (among others) the forward and reverse KL-divergence, JS-divergence, and χ^2 divergence.
- We develop a novel masking scheme allowing training of a transformer-based autoregressive model with a backspace action with no additional overhead vs MLE.
- Finally we evaluate the empirical performance of SequenceMatch-trained models, showing improved performance over the maximum likelihood objective in text generation and arithmetic.

2 PRELIMINARIES: TRAINING OBJECTIVES FOR AUTOREGRESSIVE MODELS

2.1 KL-DIVERGENCE

Typically, autoregressive models are trained against a maximum-likelihood objective. This objective can be motivated by treating our dataset as consisting of sequences of random variables $(x_1, \ldots, x_N)$, with a corresponding probability distribution $P_{\text{data}}(x_1, \ldots, x_N)$, with a fixed length N. The goal is to learn a parameterized model $P_\theta(x_1, \ldots, x_N)$ that is close to P_{data}. The KL-divergence between the data distribution and the model has a useful decomposition: $D_{\mathrm{KL}}(P_{\text{data}} \| P_\theta) = -\mathbb{E}_{x_{1:N} \sim P_{\text{data}}}\left[\sum_i^N \log P_\theta(x_i | x_{<i})\right] + C$, where C is a constant that does not depend on θ. For a dataset $\mathcal{D} = \{x_{1:N}^j\}_{j=0}^{N_{\text{data}}}$ of sequences drawn i.i.d. from P_{data}, this can be approximated with an estimator $\hat{D_{\mathrm{KL}}}(P_{\text{data}} \| P_\theta) = \frac{1}{N_{\text{data}}} \sum_j \sum_i^N \log P_\theta(x_i^j | x_{<i}^j) + C'$. Hence, minimizing the KL-divergence is equivalent to maximizing the model's (log-) likelihood of the next element in

the sequence given the previous elements. The (typically unknown) density under the data distribution P_{data} is not required. In some domains, the length of the sequences n_j differs in each example j, which can be incorporated by choosing an effective length $N = \max n_i$, and treating all sequences shorter than N as having a sequence of padding tokens appended[1]. In the sequel with some abuse of notation we will write $P_{\text{data}}(s_n)$ for $\sum_i^n \log P_\theta(x_i|x_{<i})$, for partial sequences that may not terminate until after n, and $P_{\text{data}}(x|s_n)$ to signify the probability of the next token given a partial sequence.

2.1.1 Limitations of the KL-divergence

However, while it is clear that minimizing the KL-divergence will result in $P_\theta = P_{\text{data}}$ (for a sufficiently flexible parameterization P_θ), it is not obvious what the behaviour is of models which *approximately* minimize the KL-divergence. In figure 1, a chain distribution is shown, with sequences of length N. The model P_θ has an ϵ error on each conditional, where an error leads to an OOD sequence which has no support under the data distribution. This leads to an error in the KL metric of order $n\epsilon$. However, the probability of getting to the end of the chain before an incorrect token is picked is $1 - (1 - \epsilon)^n$, and so the value of the KL-divergence is not a good metric if our main quantity of interest is how often generated sequences are in-distribution. Furthermore, the KL-divergence weights the loss by the frequency under the data distribution, leaving the model's behavior out-of-distribution from the data essentially undetermined.

In non-autoregressive generative modelling, several different divergences are commonly used, such as the Wasserstein distance (Arjovsky et al., 2017) and Fisher divergence (Song & Ermon, 2019). Particularly interesting is the χ^2 divergence $D_{\chi^2}(P_\theta, P_{\text{data}}) = \mathbb{E}_{x\sim P_{\text{data}}}\left[(P_\theta(x)/P_{\text{data}}(x) - 1)^2\right]$. Indeed, figure 1 shows that the χ^2-divergence in this case is equal to the squared probability of staying in the data distribution of sequences. We can further penalize out-of-distribution behavior by considering the divergence between mixtures $D_{\chi^2}(P_\theta, (P_{\text{data}} + P_\theta)/2)$, as we do in our practical algorithm. However, it is difficult in practice to compute any divergence involving the density of the data, which must be substituted for with an approximation from a discriminator.

In reinforcement learning, there are several methods which can minimize diverse divergences between the distribution of trajectories from an expert and a learned policy. The approaches are non-adversarial, even with unknown expert density (Garg et al., 2021; Swamy et al., 2021; Barde et al., 2020; Al-Hafez et al., 2023). A key feature of these methods is that they operate on *occupancy measures* instead of joint distributions, a concept which we describe in the next section.

3 Method

3.1 Sequence Modelling as a Markov Decision Process

We consider a sequence modelling problem represented as a Markov decision process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, with state and action spaces $\mathcal{S}, \mathcal{A}$, dynamics $\mathcal{P}(s'|s, a)$, reward function $r(s, a)$, and discount factor $\gamma \in (0, 1)$. In our case, the state space $\mathcal{S}$ is the set of all sequences (of all lengths) with elements in a finite set X, the vocabulary. The action set $\mathcal{A}$ is a finite set. For concreteness, we can assume that $X \subseteq \mathcal{A}$ (i.e. we have an `insert-token` action for each token), as well as an additional backspace action `<bkspc>`. In our case, the initial state is given by a special `<bos>` token, while the dynamics for an `insert-token` action in a state (sequence) s leads deterministically to the sequence s' consisting of s with the given token appended.

Combined with a policy $p_\theta(a|s)$, the MDP defines a distribution over (possibly infinite-length) sequences, following the generative process of sampling an action $a \sim p_\theta(\cdot|s)$, then sampling the next state $s' \sim \mathcal{P}(s'|s, a)$, etc. Finally, we assume that a special end of sequence token `<eos>` induces a terminal state: in a state s with `<eos>` as the final element, all actions cause a self-transition to s. This probabilistic process reduces exactly to the autoregressive formulation of sequence modelling when the action set is the same as the vocabulary. A backspace action in a state s deterministically transitions to a state s' with the final token in the sequence s removed.[2] An example of the states and actions can be seen in figure 2. The MDP framework formalizes the intuitive picture in figure 1: language modelling can be viewed as the traversal of a tree where the nodes are (partial) sequences.

---

[1] Some care is required, as averaging the loss of each example over its length gives an inconsistent estimator.

[2] In the case of s = `<bos>` and `<bkspc>` is used, s' = `<bos>` also

A central quantity of interest is the occupancy measure. We denote by s_t the random variable consisting of the state at time t under a policy $p(a|s)$ and the MDP defined above. Then, the occupancy measure $\rho(s,a) : \mathcal{S} \times \mathcal{A} \to [0,1]$ is the (discounted) probability of observing a particular sentence s at time t and taking action a given that sentence:

$$\rho(s,a) = (1-\gamma)p(a|s)\sum_t \gamma^t P(s_t = s). \tag{1}$$

In other words, the occupancy measure is proportional to the observed frequency of a particular (sentence, next-action) pair occurring, with occurrances discounted in time by a factor of γ. In the absence of editing actions, $\mathcal{A} = X$ and the occupancy measure is a discounted probability over (partial) sequences: for a sequence s_n of length n, $\rho_{\text{data}}(s_n, x) = (1-\gamma)\gamma^n P_{\text{data}}(s')$, where s' is the sequence obtained by appending x to s. Given editing actions which can reduce the length of a sequence, the occupancy measure becomes more complicated, as the same sequence can occur at multiple time indices. For a function r, the expectation with respect to ρ has the usual meaning: $\mathbb{E}_{(s,a)\sim\rho}\left[r(s,a)\right] = \sum_{\mathcal{S},\mathcal{A}} \rho(s,a)r(s,a)$, with sum over the discrete action space and (countably infinite) state space. Occupancy measures provide an alternative way of modelling sequences, allowing us to impose a measure over all sequences, even in the presence of editing actions. The next section illustrates that we can non-adversarially minimize a large variety of divergences between occupancy measures, compared to only the KL divergence in the typical joint probability formulation.

3.2 MINIMIZING OCCUPANCY DIVERGENCES

Our aim is to learn a policy $p_\theta(a|s)$ which induces an occupancy measure p_θ such that it is close to the data occupancy measure p_{data}. We define the data occupancy measure by forming the data policy $p_{\text{data}}(a|s)$ corresponding to the conditionals $P_{\text{data}}(x|s_n)$ and setting the probability of editing actions to zero. It is known that matching occupancy measures implies matching policies: if $\rho_\theta = \rho_{\text{data}}$ for a valid occupancy ρ_θ, then the corresponding $p_\theta(a|s) = P_{\text{data}}(a|s)$ (Syed et al., 2008). Therefore, it is reasonable to minimize divergences between occupancy measures. We extend the derivations in Garg et al. (2021) to the case with infinite-dimensional state space. We consider distances between occupancy divergences parameterized by the following form:

$$d_\psi(\rho_\theta, \rho_{\text{data}}) = \sup_{r\in\mathcal{R}} \mathbb{E}_{(s,a)\sim\rho_\theta}\left[r(s,a)\right] - \mathbb{E}_{(s,a)\sim\rho_{\text{data}}}\left[r(s,a)\right] - \psi(r), \tag{2}$$

where ψ is a convex regularizer. The critic r picks out differences between the occupancies, while if $\rho_\theta = \rho_{\text{data}}$, the difference in expectations will be zero for any r. This family of divergences includes all f-divergences such as the KL and JS-divergence, as well as the Wasserstein distance and MMD, as described in the appendix. The problem can be made tractable by adding an entropy term:

$$\inf_\theta d_\psi(\rho_\theta, \rho_{\text{data}}) - \alpha H[\rho_\theta], \tag{3}$$

with the entropy $H[\rho_\theta] = -\mathbb{E}_{(s,a)\sim\log\rho_\theta}\left[\log\rho_\theta(s,a)\right]$, and α a chosen regularization strength. Substituting in the definition of d_ψ, we obtain the min-max problem $\inf_{\rho_\theta}\sup_r L(\theta,r) = \inf_{\rho_\theta}\sup_r r\cdot(\rho_\theta - \rho_{\text{data}}) - \psi(r) - \alpha H\left[\rho_\theta\right]$. We prove in the appendix that the saddle-point property in Ho & Ermon (2016) extends to our infinite-dimensional case, so $\inf_{\rho_\theta}\sup_r L(\theta,r) = \sup_r\inf_{\rho_\theta} L(\theta,r)$. We can interpret the outer maximization as finding a critic (LeCun et al., 2006) r for sequences and actions s, a such that the model has high values on examples from the dataset and low values on the examples from the learned model. The inner minimization over θ is an entropy-regularized minimization of the KL-divergence between ρ_θ and r. Approaching this directly by explicitly learning r and ρ_θ, would lead to an objective similar to a GAN (Goodfellow et al., 2014). This is known to be difficult to train (Jabbar et al., 2020). Instead, we can solve the problem with optimization over a single variable by a transformation of variables. In the following section, we recover an objective $\mathcal{J}$ which is equivalent to the objective in equation 3, but only involves optimization over the logits of a policy. For ease of exposition, $\alpha = 1$ in the next section.

3.2.1 REFORMULATING THE OCCUPANCY DIVERGENCE MINIMIZATION PROBLEM

We first introduce the Q-function, corresponding to the discounted rewards obtained in state s by taking action a. Formally, it is the unique fixed point of the soft Bellman operator $\mathcal{B}_r^\theta$, where $\mathcal{B}_r^\theta Q(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(s,a)}\left[V^\theta(s')\right]$, for the value function $V^\theta(s) =$

$\mathbb{E}_{a\sim p_\theta(\cdot|s)}\left[Q(s,a)-\log p_\theta(a|s)\right]$. The inverse Bellman operator $\mathcal{T}^\theta$ is the inverse of this operator, given by $(\mathcal{T}^\theta Q)(s,a)=Q(s,a)-\gamma\mathbb{E}_{s'\sim\mathcal{P}(s,a)}\left[V^\theta(s')\right]$. For a fixed policy θ, there is a one-to-one correspondence between r and Q via the Bellman and inverse Bellman operators (proved in the appendix). Crucially, for the unique occupancy ρ^* which solves $\max_\theta \mathbb{E}_{s,a\sim\rho_\theta}\left[r(s,a)\right]-H[\rho_\theta]$, the optimal policy $\log p^*(a|s)$ corresponding to ρ^* is proportional to the corresponding Q-values Q^*: $\log p^*(a|s)=Q^*(s,a)-V^{\theta^*}(s)=Q^*(s,a)-\log\sum_{a'\in\mathcal{A}}\exp Q^*(s,a')$. The key idea of the following derivations is that the optimal policy is uniquely determined by the optimal Q-values, while the reward is determined by the Q-values. This allows us to optimize solely over Q-values.

Proposition 3.1. *The following equalities hold for the loss:*

$$\begin{aligned}
\inf_\theta d_\psi(\rho_\theta,\rho_{data})-H[\rho_\theta] &= \sup_r\inf_\theta \mathbb{E}_{\rho_{data}}\left[r(s,a)\right]-\mathbb{E}_{\rho_\theta}\left[r(s,a)\right]-H[\rho_\theta]-\psi(r),\\
&= \sup_Q\inf_\theta \mathbb{E}_{\rho_{data}}\left[\mathcal{T}^\theta Q\right]-\mathbb{E}_{\rho_\theta}\left[(\mathcal{T}^\theta Q)\right]-H[\rho_\theta]-\psi(\mathcal{T}^\theta Q),\\
&= \sup_Q\inf_\theta \mathbb{E}_{\rho_{data}}\left[\mathcal{T}^\theta Q\right]-(1-\gamma)\mathbb{E}_{\mathcal{P}_0}\left[V^\theta(s_0)\right]-\psi(\mathcal{T}^\theta Q),\\
&= \sup_Q\inf_\theta \mathbb{E}_{\rho_{data}}\left[\phi(Q(s,a)-\gamma\mathbb{E}_{\mathcal{P}}\left[V^\theta(s')\right])\right]-(1-\gamma)\mathbb{E}_{\mathcal{P}_0}\left[V^\theta\left(s_0\right)\right],\\
&= \sup_Q\inf_\theta \mathbb{E}_{\rho_{data}}\left[\phi(Q(s,a)-\gamma\mathbb{E}_{\mathcal{P}}\left[V^\theta(s')\right])\right]-\mathbb{E}_{\rho}\left[V^\theta(s)-\gamma V^\theta(s')\right],\\
&= \sup_Q \mathbb{E}_{\rho_{data}}\left[\phi(Q(s,a)-\gamma\mathbb{E}_{\mathcal{P}}\left[V(s')\right])\right]-\mathbb{E}_{\rho}\left[V(s)-\gamma V(s')\right],
\end{aligned}$$

where ϕ is concave, $\mathbb{E}_{\rho_{data}}$ denotes expectations over sampled states and actions s,a, $\mathbb{E}_{\mathcal{P}}$ denotes an expectation over successor states s', and $\mathbb{E}_\rho$ denotes an expectation over sampled states s and successor states s', for any occupancy ρ. $V(s)$ (without θ) is given by $V(s)=\log\sum_{a'\in\mathcal{A}}\exp Q(s,a')$.

Proof. The full proof is given in the appendix. As a sketch, the first equality holds from the previous section. The second is obtained by replacing r with $\mathcal{T}^\theta Q$ and verifying that the two optimization problems are equal. The third line is via a telescoping sum argument first described in (Kostrikov et al., 2019). In the fourth line we replace $\psi(r)$ with a simpler regularizer $\mathbb{E}_{s,a\sim\rho_{\text{data}}}\left[g(r(s,a))\right]$, where $g(r)=r-\phi(r)$ if $r\in\Omega$, and infinity otherwise. The fifth line follows from expanding the telescoping sum in a different way, incorporating samples from any policy. In the final line we parameterize the policy from the Q-values, setting $\log p_Q(a|s)=Q(s,a)-\log\sum_{a'\in\mathcal{A}}\exp Q(s,a')$. We then show that the optimization problem over (Q,p_Q) has the same optimum as the optimization over (Q,θ), so we can optimize solely over Q. $\square$

We relabel Q as ℓ_θ to make the connection to logits clearer, resulting in the fully supervised objective over logits ℓ_θ: $\mathcal{J}(\ell_\theta)=\frac{1}{\alpha}\mathbb{E}_{\rho_{\text{data}}}\left[\phi(\alpha\ell_\theta(a|s)-\alpha\gamma V(s')\right]-\frac{1}{2}\mathbb{E}_{\rho_{\text{data}}}\left[V(s)-\gamma V(s')\right]-\frac{1}{2}\mathbb{E}_{\rho_\theta}\left[V(s)-\gamma V(s')\right]$, where $(s,a,s')\sim\rho$ corresponds to sampling s,a from ρ and s' from $\mathcal{P}(\cdot|s,a)$, and $V(s)=\log\sum_{a'\in\mathcal{A}}\exp\ell_\theta(a'|s)$. Minimizing this objective is equivalent to $\min_\theta d_\psi(\rho_\theta,\rho_{\text{data}})-\alpha H[\rho_\theta]$, where $d_\psi(P,Q)=\sup_{r\in\Omega}\mathbb{E}_{x\sim P}\left[\phi(r(x))\right]-\mathbb{E}_{x\sim Q}\left[r(x)\right]$. By choosing Ω and ϕ, we can recover f-divergences, including KL, JS and χ^2 divergences, and additionally the Wasserstein and MMD distances. The corresponding choices are given in the appendix.

4 PRACTICAL OCCUPANCY MATCHING WITH AUTOREGRESSIVE MODELS

In practice, we wish to train a parameterized model $p_\theta(a|s)$ which can serve as a policy, emitting a probability distribution over the next action given a partially completed sequence s. A typical choice is a transformer (Vaswani et al., 2017): with parameters θ it gives a distribution over the next token x_i given the previous tokens $x_{<i}$, parameterized as unnormalized logits ℓ_θ. Thus the MLE loss, with a complete sequence $x_{1:N}$, can be written as $\hat{\mathcal{L}}_{\text{MLE}}(\ell_\theta)=\sum_{i=1}^N\ell(x_i|x_{<i})-\log\sum_{x'\in X}\exp\ell(x'|x_{<i})$, or $\hat{\mathcal{L}}_{\text{MLE}}(\ell_\theta)=\sum_{i=1}^N\ell(x_i|x_{<i})-V(x_{<i})$.

To form an estimator for the loss derived in the previous section, samples from ρ_θ are required. We obtain these samples by sampling complete sequences from the policy autoregressively and weighting the partial sequence at time t by a factor of γ^t. We similarly sample sequences from ρ_{data}

by sampling complete sequences from P_{data} and weighting. So, for a length-N sequence of states $s_{1:N}$ from a dataset, corresponding actions $a_{1:N}$ and a generated length-M sequence $u_{1:M}$ of states from the model, we can form an estimator for the loss from the previous section:

$$\hat{\mathcal{J}}(\ell_\theta) = \underbrace{\sum_i^N \gamma^i \frac{1}{\alpha}\phi\left(\alpha\ell_\theta(a_i|s_i) - \gamma\alpha V(s_{i+1})\right)}_{\text{Penalized difference from action logit to next state value}} - \underbrace{\sum_i^N \frac{\gamma^i}{2}\left[V(s_i) - \gamma V(s_{i+1})\right]}_{\text{State, next state value difference under data}}$$
$$- \underbrace{\sum_i^M \frac{\gamma^i}{2}\left[V(u_i) - \gamma V(u_{i+1})\right]}_{\text{State, next state value difference under model}} + \underbrace{\frac{\gamma^N}{\alpha(1-\gamma)}\phi\left(\alpha(1-\gamma)V(s_N)\right) - \frac{\gamma^N}{2}V(s_N) - \frac{\gamma^M}{2}V(u_M),}_{\text{Loss from completed sequences}} \tag{4}$$

and $V(s) = \log\sum_{a'\in\mathcal{A}}\exp\ell_\theta(a'|s)$. The separate treatment of the <eos> tokens arises from taking the sum over the infinite timesteps in equation 1 in the terminal states. As shown in the appendix, the estimator is unbiased and consistent for finite ϕ. It has also been shown (Al-Hafez et al., 2023) that minimizing the mixture divergence $D_{\chi^2}(\rho_{\text{data}}, (\rho_{\text{data}} + \rho_\theta)/2)$ is more effective than simply minimizing the χ^2-divergence between model and data. This can be implemented by calculating the loss for the χ^2-divergence (with $\phi(x) = x - \frac{1}{4}x^2$) and adding an additional regularization term $\mathbb{E}_{s,a,s'\sim\rho_\theta}\left[(\alpha\ell_\theta(a|s) - \gamma\alpha V(s'))^2\right]$. We show in the appendix that with no backspace actions, $\lim_{\alpha\to 0}\mathcal{J}_{\ell_\theta} = \mathrm{D}_{\text{KL}}(\rho_{\text{data}}\|\rho_\theta)$ reduces to a γ-reweighted MLE objective.

4.1 EFFICIENT TRAINING

Editing actions which can delete previous parts of the input are challenging to implement while retaining the fast training of transformer-based autoregressive models. For instance, the sequence of actions `[a; b; <bkspc>]` cannot be fed directly into a policy network $p_\theta(a|s)$, since it contains actions, not states. The sequence `[a; b; <bkspc>]` is not a valid state: the corresponding state is `[<bos> a]`. In order to convert this into a form where we can compute the relevant logits using masked attention, we must pre-process the sequence of actions into corresponding inputs, labels, masks and position IDs using algorithm A in the appendix. The preprocessing is illustrated in figure 2. On the other hand, generation with backspace actions is straightforward: we already keep previous key-value cached values for generation with transformers. When <bkspc> is sampled, we simply roll back the state of the key-value cache and position id with negligible overhead. Additionally, the loss requires sampling from the model during training, which is typically slow. However, the sequences do not need to be exactly sampled from the current policy. Since any policy can be used, sequences generated from the policy at previous training steps are stored in a replay buffer (Mnih et al., 2013) and reused. We give an empirical analysis of the overhead when using SequenceMatch in the appendix.

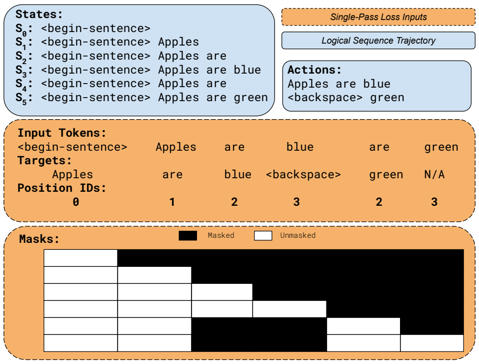
States:
S0: <begin-sentence>
S1: <begin-sentence> Apples
S2: <begin-sentence> Apples are
S3: <begin-sentence> Apples are blue
S4: <begin-sentence> Apples are
S5: <begin-sentence> Apples are green
Single-Pass Loss Inputs
Logical Sequence Trajectory
Actions:
Apples are blue
<backspace> green
Input Tokens:
<begin-sentence> Apples are blue are green
Targets:
Apples are blue <backspace> green N/A
Position IDs:
0 1 2 3 2 3
Masks:
Masked
Unmasked

Figure 2: Transforming states and actions to single-pass inputs for parallel training.

4.2 AUGMENTING EXPERT SEQUENCES WITH BACKSPACE

To provide the policy with examples of how the <bkspc> action should be used, we augment the data sequences as follows: with (small) probability η, we replace a sequence $\ldots, x_{i-1}, x_i, x_{i+1}, \ldots$ with x_{i-1}, x_i, x'_i, <bkspc>, $x_{i+1}, \ldots$, where x'_i is chosen randomly from the vocabulary. However, we keep the action at position i as x_{i+1}, with the result that the overall MDP is augmented with a stochastic dynamics: with probability η a random token is inserted, instead of the chosen action. This is also applied to sequences which exceed the context length: the action is kept the same but the next token is forced to be the <eos> token. This introduces bias, as the policy learns to match the data distribution under a slightly different MDP than generation takes place in. In practice however, it leads to improved performance compared to the policy learning with no examples of <bkspc>.

Algorithm 1: Training an autoregressive model against a SequenceMatch objective

```
Input : Dataset $\mathcal{D}$ of data sequences, gradient-based optimizer step, number of train steps $n_{\text{train}}$,
        parameters $\alpha, \beta, \gamma, \phi$, sampling interval $k_{\text{sample}}$, fixed context length $T$
Add noise and process data sequences with algorithm A to form new effective trajectories
Initialize buffer $\mathcal{B}$ of model sequences; Initialize autoregressive policy $\ell_\theta(\cdot|s)$
for $k$ in $n_{train}$ do
    if $k \mod k_{sample} = 0$ then
        Populate $\mathcal{B}$ with trajectories $\mathcal{T} \sim \ell_\theta$; Process added sequences with algorithm A
        Remove oldest model sequences from $\mathcal{B}$
    end
    Sample dataset trajectories $\mathcal{T}_{\text{data}} \sim \mathcal{D}$ and model trajectories $\mathcal{T}_{\text{model}} \sim \mathcal{B}$
    Compute $g = \nabla_\theta \hat{\mathcal{J}}(\ell_\theta, \alpha, \gamma, \mathcal{T}_{\text{data}}, \mathcal{T}_{\text{model}})$ and update $\theta$ via step using gradient $g$
end
```

5 RELATED WORK

Text Degeneration in Large Language Models

In natural language processing the phenomenon of *text degeneration* can occur, when a language model produces repetitive or nonsensical sequences (Holtzman et al., 2019). Many explanations have been proposed to explain this phenomenon (Fu et al., 2021; Welleck et al., 2019); a leading theory is that the large vocabulary size induces the model to over-estimate the probability of OOD tokens. Once these tokens are sampled, the model's context is now out-of-distribution. Measures to mitigate this problem include top-k sampling (Fan et al., 2018), restricting generations to the k most likely tokens, and top-p sampling (Holtzman et al., 2019), an adaptive variant of top-k sampling. In addition, alternative training measures have been proposed to reduce the probability of the model producing OOD tokens. Unlikelihood training (Welleck et al., 2019) is discussed in detail in the appendix, while contrastive methods have also been proposed (Jiang et al., 2022), which force the representations of repetitive text to be far from the representations of correct text.

Matching Divergences in Imitation Learning

In the imitation learning(Ng & Russell, 2000) subfield of RL, the objective is to learn a policy giving a distribution over actions in each state, such that the distribution over trajectories is close to distribution of provided expert trajectories. A simple approach is behavioral cloning (Esmaili et al., 1995), which maximizes the likelihood of the expert's chosen actions, on average over the states that the expert is in. However, it has been shown (Ross et al., 2011) that this approach results in a *compounding error* problem, where the further the trained model gets from the typical expert states, the worse the model performs, incurring increasing error. Ho & Ermon (2016) show that minimizing the occupancy divergence between the expert and a learned policy could be written as a two-variable saddle-point problem. This more sophisticated method can take the dynamics of the problem into account, learning a policy which can return to the typical expert states if it erroneously leaves them. In Garg et al. (2021), this was further developed via a change of variables to only require a non-adversarial optimization over one variable. We can view our approach as a specialization of the IQ-Learn algorithm in Garg et al. (2021) to autoregressive sequence models.

6 EXPERIMENTS

In this section we demonstrate that SequenceMatch can be used with large, state-of-the-art models, and that it can be useful for downstream applications. The experiments also allow some insight into the relative importance of the different components of SequenceMatch, namely the `<bkspc>` token and the alternative loss. The first experiment shows that SequenceMatch can improve accuracy on a downstream task, and that it can detect OOD states. The second experiment shows that SequenceMatch training can generate sequences with higher similarity to the data compared to a variety of baselines. In the appendix, section G, we describe three additional experiments, on translation, multiplication, and prime factorization. In all experiments, we finetune Llama2-7b (Touvron et al., 2023), using quantized low-rank adapters (Dettmers et al., 2023). In addition to the adapters, a row is added to the unembedding layer for the `<bkspc>` token, and the unembedding layer is trained.

6.1 ARITHMETIC

We first examine the potential for SequenceMatch to improve the performance on downstream tasks. The dataset is the arithmetic add-or-sub-in-base sub-task of the math-dataset (Saxton et al., 2018), consisting of questions such as `In base 7, what is -1240 - -4156?`. We compare a maximum-likelihood model, a behavioral cloning model, and a SequenceMatch model, with varying levels and types of noise. 'Random noise' is generated by sampling a token at random from the vocabulary, and hence is not very likely to be a common mistake made when solving the problem. 'Ground-truth noise' is generated by sampling a token from the set $\{0, \ldots, b-1\}$, where b is the base in the question. The latter type of noise is expected to generate sequences that are far closer to the type of inaccuracies that a model (or human) might make while solving the problem. However, the random noise is generic to all tasks, while the ground-truth noise must be constructed for a given task manually. Both types of noise are only added to the solution digits, not the question digits. We use a small dataset of only 5,000 questions, to demonstrate the improved generalization abilities of SequenceMatch. The prompts for the SequenceMatch generations are taken from the training dataset and truncated at the end of the question. The accuracy is computed over a held-out test set of 200 questions, and error bars obtained by training two models with different random seed.

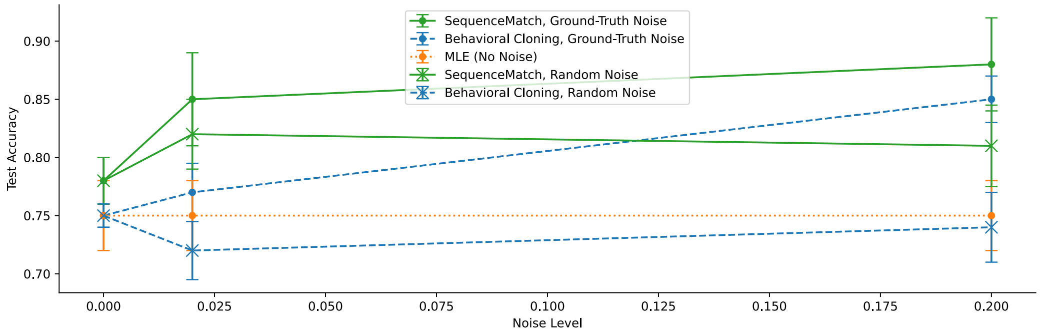

Figure 3: Accuracy on the arithmetic task against noise level (frequency with which noise tokens are added), for ground-truth noise consisting of digits and random noise consisting of random tokens. The ground-truth noise improves accuracy for the behavioral cloning and SequenceMatch models. The random noise does not improve performance for the behavioral cloning model, but does somewhat for the SequenceMatch model, likely helping the model to learn the dynamics of `<bkspc>`.

6.1.1 RESULTS

The results are shown in figure 3. It is clear that the models trained under the SequenceMatch loss outperform both the behavioral cloning (BC) and MLE baselines. With zero added noise, the improvement is small, as the model is not able to learn the backspace dynamics with no demonstrations from the expert. However, with a small amount of added noise, the SequenceMatch model has demonstrations of the use of `<bkspc>` and can start learning the dynamics through sampling. The BC model improves performance at higher levels of ground-truth noise, as the high level of noise acts as strong data augmentation. Indeed, this improved performance seems roughly proportional to the amount of added noise, as expected. Similarly, while random noise is useful to the SequenceMatch model due to the dynamics information, it does not help the BC model.

Qualitatively, the SequenceMatch-trained model is able to detect when it is out-of-distribution very accurately. In table 1 the best-performing SequenceMatch model is prompted with partial solutions which are incorrect (the first four in the test set with added random digit tokens). In each case it is able to detect the OOD case and return closely to the correct answer.

6.2 TEXT GENERATION

We use the same model and architecture as the previous section. Sequences are drawn from the openwebtext dataset[3], an open-sourced dataset similar to the training set for GPT-2 (Radford et al., 2018),

---

[3] https://github.com/jcpeterson/openwebtext

Ground Truth QA	**Prompt with Mistake**	**Completion Actions**	**Final State**
`In base 2, what is -1110111011001100 + 10100? Solution:  -1110111010111000`	`In base 2, what is -1110111011001100 + 10100? Solution:  -1011`	`<bkspc><bkspc><bkspc> 1101111<bkspc>1011000010 <eos>`	`-1110111011000010`
`In base 8, what is 4354 + 33? Solution:  4407`	`In base 8, what is 4354 + 33? Solution:  5`	`<bkspc>4417<eos>`	`4417`
`In base 8, what is -4 + -576122?  Solution:  -576126`	`In base 8, what is -4 + -576122?  Solution:  -374`	`<bkspc><bkspc><bkspc> 576126<eos>`	`-576126`
`In base 5, what is 10 - 3311121?  Solution:  -3311111`	`In base 5, what is 10 - 3311121?  Solution:  -31`	`<bkspc>31<bkspc>11111<eos>`	`-3311111`

Table 1: Mistake-conditioned completions for the arithmetic task. We add a random set of digit tokens to the prompt and generate from the SequenceMatch-trained model. The SequenceMatch model correctly deletes the initial tokens in all cases and eventually generates the correct solution in three of four cases.

Model	MLE	MLE + C.S	MLE + ULK	MLE + `<bkspc>`	SequenceMatch
MAUVE (↑)	0.85 ± 0.03	0.86 ± 0.03	0.89 ± 0.02	0.84 ± 0.02	**0.91** ± **0.02**
n-gram $\mathbb{H}$ (↑)	4.57 ± 0.02	4.43 ± 0.02	4.57 ±0.01	4.59 ± 0.01	**4.60** ± **0.01**
Diversity (↑)	0.56 ± 0.02	0.35 ± 0.03	**0.57** ± **0.01**	0.56 ± 0.01	0.56 ± 0.01
Perplexity (↓)	**6.99** ± **0.02**	N/A	7.10 ± 0.02	7.02 ± 0.02	7.13 ± 0.03

Table 2: A Llama-2-7b model fine-tuned on the openwebtext dataset with different training and sampling regimes. Error bars are over two draws of 1000 evaluation samples. C.S. and ULK are contrastive sampling and unlikelihood loss training, respectively. The SequenceMatch model achieves the highest MAUVE score and n-gram entropy, with diversity very close to the best value from the MLE + unlikelihood training.

with a 1024-length context window, truncating sequences that are longer. The model-generated trajectories are generated from examples from the training dataset, with a prompt length randomly chosen with a maximum of 256. The generated sequences have a max length of 512 (although they may terminate earlier). We compare a SequenceMatch-trained model against several baselines: a model trained against the typical MLE objective, and a behavioral cloning model trained with injected noise and `<bkspc>` labels. We also test `MLE + C.S.`, which is MLE with contrastive sampling (Jiang et al., 2022). Finally, `MLE + ULK` is maximum-likelihood with the unlikelihood token loss (Welleck et al., 2019). We train for 5,000 gradient steps. The SequenceMatch parameters are set to $\alpha = 0.01$, $\eta = 0.001$ and $\gamma = 0.998$. Our main metric for quality of generations is the MAUVE score (Pillutla et al., 2022), a non-parametric method for evaluating the quality of a generative model. The MAUVE score is formed by taking a low-dimensional PCA of an embedding of the generated sequences. The score is a mixture of forward and reverse KLs between data and model-generated sequences, between zero and one (higher is better). Additionally we report the n-gram entropy and the diversity metric (Jiang et al., 2022), given by $\prod_{n=2}^{4}\left(1.0 - \frac{\text{rep}-\text{n}}{100}\right)$, where rep-n $= 100 \times \left[1.0 - \frac{|\text{unique n-grams }(\hat{\boldsymbol{x}})|}{|\text{total n-grams }(\hat{\boldsymbol{x}})|}\right]$ for a generation $\hat{\boldsymbol{x}}$.

6.2.1 RESULTS

Table 2 shows that the SequenceMatch-trained model is able to achieve higher MAUVE score compared to the baselines. It also improves with respect to the n-gram entropy. On the diversity metric, all models are similar, except the contrastive sampling model. The SequenceMatch-trained model is outperformed on the perplexity metric by the BC and MLE-trained methods. This is expected, as the training objective for BC and MLE is exactly the log-perplexity. However, on the MAUVE score, only unlikelihood and SequenceMatch offer a substantial improvement to MLE. Of course, unlikelihood relies on a heuristic to penalize repetitions: a heuristic not appropriate in e.g. arithmetic.

7 CONCLUSION

We address the compounding error problem in autoregressive sequence generation by formulating the problem in the imitation learning framework, deriving a general non-adversarial objective for minimizing divergences between occupancy measures induced by a learned model and the data distribution. We develop a novel masking scheme to train a transformer-based autoregressive model with a backspace action with small overhead vs MLE, further reducing compounding error by allowing backtracking. Empirically, the SequenceMatch objective leads to improvements over MLE at text generation and arithmetic. Future work could investigate how qualities of generations change with choice of divergence, or find methods to reduce the overhead of the SequenceMatch objective.

8 ACKNOWLEDGEMENTS

This research was supported by funding from the following: Stanford HAI, NSF(#1651565), ARO (W911NF-21-1-0125), ONR (N00014-23-1-2159), and the CZ Biohub. We thank Anuj Nagpal, Div Garg and Andy Shih for valuable discussions and feedback on this research direction.

REFERENCES

Firas Al-Hafez, Davide Tateo, Oleg Arenz, Guoping Zhao, and Jan Peters. LS-IQ: Implicit reward regularization for inverse reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223. PMLR, 2017.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why Exposure Bias Matters: An Imitation Learning Perspective of Error Accumulation in Language Generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 700–710, 2022.

Paul Barde, Julien Roy, Wonseok Jeon, Joelle Pineau, Chris Pal, and Derek Nowrouzezahrai. Adversarial soft advantage fitting: Imitation learning without policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 12334–12344, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Nasser Esmaili, Claude Sammut, and GM Shirazi. Behavioural cloning in control of a dynamic system. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 3, pp. 2904–2909. IEEE, 1995.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, 2018.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. A theoretical analysis of the repetition problem in text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12848–12856, 2021.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. IQ-Learn: Inverse soft-Q learning for imitation. In *NeurIPS*, 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Neural Information Processing Systems (NeurIPS)*, 2014.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

Abdul Jabbar, Xi Li, and Bourahla Omar. A Survey on Generative Adversarial Networks: Variants, Applications, and Training. *arXiv:2006.05132 [cs]*, June 2020.

Shaojie Jiang, Ruqing Zhang, Svitlana Vakulenko, and Maarten de Rijke. A simple contrastive learning objective for alleviating neural text degeneration. *arXiv preprint arXiv:2205.02517*, 2022.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations*, 2019.

Dohyun Kwon, Yeoneung Kim, Guido Montúfar, and Insoon Yang. Training Wasserstein GANs without gradient penalties. *arXiv:2110.14150 [cs, math]*, October 2021.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A Tutorial on Energy-Based Learning, 2006.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, 2013.

Andrew Y Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *In Proc. 17th International Conf. on Machine Learning*. Citeseer, 2000.

Krishna Pillutla, Lang Liu, John Thickstun, Sean Welleck, Swabha Swayamdipta, Rowan Zellers, Sewoong Oh, Yejin Choi, and Zaid Harchaoui. MAUVE scores for generative models: Theory and practice. *arXiv preprint arXiv:2212.14578*, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2018.

Douglas Rizzolo and Francis Edward Su. A fixed point theorem for the infinite-dimensional simplex. *Journal of mathematical analysis and applications*, 332(2):1063–1070, 2007.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, pp. 627–635, 2011.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2018.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. Toward Diverse Text Generation with Inverse Reinforcement Learning. *arXiv:1804.11258 [cs, stat]*, June 2018.

Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(4):171–176, 1958.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Gokul Swamy, Sanjiban Choudhury, Zhiwei Steven Wu, and J. Andrew Bagnell. Of Moments and Matching: Trade-offs and Treatments in Imitation Learning. *arXiv:2103.03236 [cs, stat]*, March 2021.

Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1032–1039, 2008.

Shichang Tang. Lessons Learned from the Training of GANs on Artificial Datasets. *arXiv:2007.06418 [cs, stat]*, July 2020.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2019.