Improving Neural Machine Translation with Offline Evaluations

Minkyung Park and **Byung-Jun Lee**
Department of Artificial Intelligence
Korea University, Republic of Korea
{swwwjkl538,byungjunlee}@korea.ac.kr

Abstract

Reinforcement learning (RL) offers a principled framework for optimizing a given reward function and has been applied to a neural machine translation (NMT) problem to maximize arbitrary task metrics. However, previously adopted RL algorithms for NMT (e.g., policy gradient) are generally slow as they require online data collection, and limits the algorithm's applicability to specific reward functions that can be evaluated online. In this paper, we present an offline RL algorithm called CER (conservative expectile regression). Despite the demanding nature of offline RL tasks, which are even more difficult with large models, this algorithm is capable to learn stably by explicitly exploiting the properties of NMT's RL formulation, such as the deterministic transition function. We analyze and discuss the design choices of CER, and demonstrate in the experiments that the proposed method outperforms its competitors for offline reward optimization in NMT.

1 Introduction

Standard training of neural machine translation (NMT) systems relies on the maximum likelihood estimation (MLE) with ground-truth parallel corpus , where we assume that every single instance in the dataset is a correct translation. Despite the remarkable progress made so far (Lewis et al., 2020; Brown et al., 2020), there is always a need of training methods that can learn without the perfect supervision, e.g., learning from noisy data (Guo et al., 2021), or human feedback (Nguyen et al., 2017).

To this end, reinforcement learning (RL) has gained attraction to train models beyond the standard MLE training, as it offers a principled framework to optimize for the given reward function. In the applications of NMT (or text generation in general), it has been widely researched to mainly reduce the *exposure bias* (Wang and Sennrich, 2020) and/or optimize for the non-differentiable metrics like BLEU (Edunov et al., 2018). However, most of the previous works have focused on designing online RL algorithms, which possess inherent weaknesses in a number of practical perspectives. Primarily, online RL algorithms are slow due to the low probability of obtaining good samples during online data collection, and is not possible if the reward/transition functions cannot be sampled online, e.g., learning on human feedback.

Recently, there was a large amount of research to develop efficient and stable offline reinforcement learning algorithms that only utilize previously collected data (Kumar et al., 2020), and some of them have achieved small successes on real-world tasks like robot manipulations (Mandlekar et al., 2021). While offline learning is free from the inherent weaknesses of online learning, it adds another difficult algorithmic challenge of *distribution shift* on top of the challenges and destabilizes learning. We indeed empirically observed that typical offline RL algorithms with bootstrapped action-value learning become easily unstable in NMT, and less performant in practice. Nevertheless, we also show that there are clear limitations in optimizing the policy with the importance sampling alone, which implies that action-value learning should not be abandoned.

In this paper, we aim to develop an efficient offline RL algorithm that is able to handle all of the above challenges. To this end, we explicitly exploit the unique characteristics that the RL formulation of NMT has: deterministic transition, and the fact that choosing different actions will never lead to the same state in the future. These characteristics enable us to use the proposed algorithm, conservative expectile regression (CER), which is specialized to have high efficiency in this particular setting. We also present a way to extract a policy and decode from the learned action-values to obtain the highest performing translation possible. In the experiments, we demonstrate the performance of the CER algorithm by designing an offline RL experi-

Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 385–397
November 1–4, 2023.

ments based on various IWSLT'14 and WMT'14 parallel datasets. We show that CER outperforms other offline RL algorithms while it only requires a minor change in architecture and small amount of extra computation that parallelizes.

2 Preliminaries

2.1 RL formulations for NMT

The goal of neural machine translation (NMT) is to produce a translation $\mathbf{y} = (y_0, ..., y_T)$ in a target language given a sentence $\mathbf{x} = (x_0, ..., x_S)$ in a source language. Each $y, x \in V$ are tokens from a finite vocabulary set V.

In this work, we model the neural machine translation task as a Markov Decision Process (MDP), where the system state at each time step t is $\mathbf{s}_t = (\mathbf{x}, \mathbf{y}_{<t})$, the partial translation generated so far. The action corresponds to choosing the next token from a finite action space, the vocabulary set: $a_t = y_t \in V$. The machine translator corresponds to a policy $\pi(a_t|\mathbf{s}_t) = p(y_t|\mathbf{y}_{<t}, \mathbf{x})$. After the translator chooses a token, it receives a reward $r_t = R(\mathbf{s}_t, a_t)$ and a next state $\mathbf{s}_{t+1}$, with a deterministic transition rule $\mathbf{s}_{t+1} = (\mathbf{x}, \mathbf{y}_{<t+1})$ with $\mathbf{y}_{<t+1} = (\mathbf{y}_{<t}, a_t)$. This procedure will produce a trajectory (i.e., a full translation) $\tau = (\mathbf{s}_0, a_0, r_0, ..., \mathbf{s}_T, a_T, r_T)$ given a policy.

We also define the return as the sum of discounted rewards from time-step t, $G_t = \sum_{t'=t}^{T} \gamma^{t'} r_{t'}$ where $\gamma \in (0, 1]$ is a discount factor. The goal is then to find a policy that maximizes the expected return at the initial state, $J(\pi) = \mathbb{E}_{\tau\sim\pi}[G_0]$. To do this, the action-value function of policy is often computed and used: $Q^\pi(\mathbf{s}, a) = \mathbb{E}_{\tau\sim\pi}[G_0|\mathbf{s}_0 = \mathbf{s}, a_0 = a]$, which is the expected return following π, starting from taking action a at state $\mathbf{s}$.

One particular aspect of NMT as an RL problem is that most of the task metrics we use for the reward function can only evaluate the completed translations. In these cases, the nonzero reward will only be assigned to the end of the trajectory, i.e. $r_t = 0, \forall t < T$. We assume the case throughout this paper.

Learning from offline data In contrast to online RL algorithms, offline RL uses previously collected dataset $\mathcal{D}$ without any additional data collection. In terms of NMT, offline RL is similar to the usual supervised learning of NMT methods, but it enables us to optimize arbitrary score metrics. In the following, with a slight abuse of notations, we also refer $\mathcal{D}$ as an empirical distribution induced by dataset $\mathcal{D}$, i.e., $\mathcal{D}(a|\mathbf{s}) = \frac{\sum_{(\mathbf{s}_i, a_i)\in\mathcal{D}} \mathbf{1}_{\mathbf{s}_i=\mathbf{s}, a_i=a}}{\sum_{\mathbf{s}_i\in\mathcal{D}} \mathbf{1}_{\mathbf{s}_i=\mathbf{s}}}$ is the behavior policy estimated from $\mathcal{D}$.

3 The Offline RL Framework for NMT

In this section, we discuss the weaknesses of the previous offline RL methods (§3.1) and introduce a practical algorithm, conservative expectile regression (CER), that avoids those issues and enables stable training of action-value Q (§3.2, §3.3). Then, we describe how we can obtain a highly rewarding translation sample from learned Q (§3.4).

3.1 Challenges in Offline RL on NMT

While RL in principle provides an effective framework for optimizing neural machine translators on arbitrary metrics, due to the large state-action space and large language models, designing an offline RL algorithm that works faces a number of challenges. We discuss below the weaknesses of various candidates that we did not choose to develop.

Policy gradient The most widely used algorithm for NMT (or text generation in general) is policy gradient (PG, Ranzato et al., 2016; Wu et al., 2018; Choshen et al., 2020; Kiegeland and Kreutzer, 2021). PG is an on-policy algorithm, meaning that the samples from the current policy π are required to optimize the policy. To apply PG in an offline setting, we need to apply an importance sampling:

$$\begin{aligned}\nabla J(\pi) &= \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} G_t \nabla \log \pi(a_t|\mathbf{s}_t)\right] \\ &= \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T} w_t G_t \nabla \log \pi(a_t|\mathbf{s}_t)\right],\end{aligned}$$

with importance weights $w_t = \prod_{t'=t}^{T} \frac{\pi(a_{t'}|s_{t'})}{\mathcal{D}(a_{t'}|s_{t'})}$.

Note that computing the importance weight w_t requires multiplying per-timestep importance weight over a number of timesteps. It can be easily seen that such an estimator suffers from a variance that grows exponentially as the trajectory gets longer. Furthermore, for the usual parametrized policy π with a softmax head, the probability of any action is nonzero. This implies that the dataset $\mathcal{D}$ should cover all possible generations, which is simply not possible.

Approximated PG To overcome such issues of PG, Pang and He (2021) proposes GOLD, which

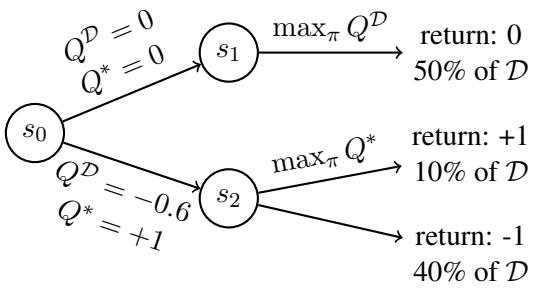

Figure 1: An illustrative example of the suboptimality of "single-step" methods, which cannot take better future decisions into account. Although the best policy would be choosing to go to s_2 to get a return of 1. maximizing $Q^{\mathcal{D}}$ leads to s_1 because of the bad data distribution after s_2. See (Snell et al., 2022) also for other examples.

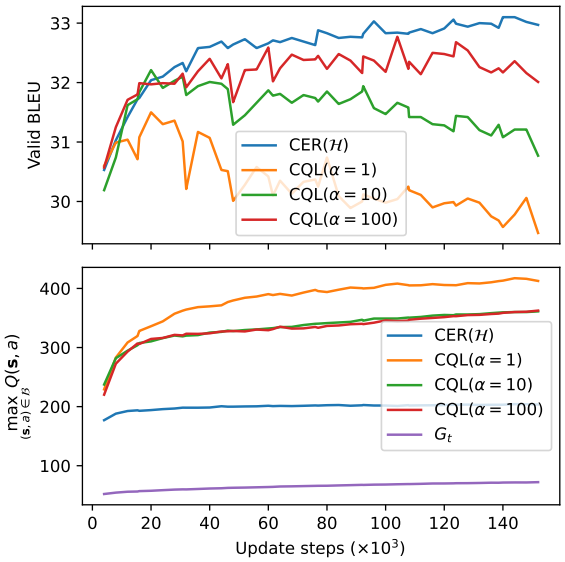

Figure 2: A learning curve on BLEU of validation data of IWSLT14 de-en task, showing a BLEU score of valid data and a maximum action-value among a batch. Note that a maximum of return G_t/true action-value Q is upper bounded by 100. It is shown that using a bootstrapped update, i.e. CQL, leads to the divergence of Q value without a careful tuning of α. Learning curves of ILQL are not presented here, but they also have similar trends to that of CQL. On the other hand, CER is much less sensitive to α (we use $\alpha = 1$ for CER throughout all the experiments, see §4.2)

is an approximation of PG that uses truncated importance weight $w_t \approx \pi(a_t|s_t)$ assuming uniform probability for samples in a dataset. This choice has two implications; first, by truncating the importance weight after the current timestep, it can be equivalently seen as maximizing the behavior value, $Q^{\mathcal{D}}$. This will make the algorithm work like the "single-step" update methods (Snell et al., 2022), which estimate the value of the behavior policy, and act greedily according to the behavior value $Q^{\mathcal{D}}$ instead of maximizing the sum of rewards. As shown in Figure 1, single-step update methods may result in a myopic, suboptimal policy.

Secondly, as we assume uniform distribution over sampled actions in a dataset, the gradient estimate becomes biased as most of the actions will not be sampled despite their nonzero probability. The algorithm will act as trajectories that are not sampled have the return of 0, and this makes the algorithm dependent on the signs of returns. For example, if all returns G_t are negative, then the probability of any trajectory in the dataset to be sampled by the policy π will be updated toward zero. These implications of approximated PG encourage us to design an algorithm that learns a non-behavior action-value, i.e., multi-step methods, to avoid the limitations.

Bootstrapped update of Q The standard way of learning Q^π in recent deep RL algorithms is using a bootstrapped update according to the following temporal difference (TD) loss:

$$\mathcal{L}_{TD} = \frac{1}{2}\mathbb{E}_{\mathcal{D}}[(r + \gamma \mathbb{E}_{\pi(a'|\mathbf{s}')}\hat{Q}(\mathbf{s}', a') - Q(\mathbf{s}, a))^2]$$

where $\hat{Q}$ is a target network and the experience tuple $(\mathbf{s}, a, r, \mathbf{s}')$ is sampled from the dataset $\mathcal{D}$. We can also simultaneously do the policy optimization by using $\max_{a'} \hat{Q}(s', a')$ instead of $\mathbb{E}_{\pi(a'|\mathbf{s}')}\hat{Q}$, which leads to a Q-learning algorithm that achieves Q^*. However, when dealing with large models, this bootstrapped update introduces a number of technical difficulties:

- Requires doubled number of parameters due to the target network, and the target network update rate should be tuned.
- Under the sparse reward of NMT, it takes T times of updates to propagate the rewards to the Qs of initial states.
- When combined with an offline setting, $\mathcal{L}_{TD}$ uses $\hat{Q}(\mathbf{s}', a')$ that may have not been trained, and is prone to divergence when π is very different from data collection policy (Kumar et al., 2019). A number of different offline RL algorithms have been proposed to prevent the divergence, but these typically require another hyperparameter to be tuned to tradeoff the possibility of divergence and the possible performance improvement.

- It may lead to poor generalization and degenerate feature representation (Kumar et al., 2021) that can harm performances on new source sentences that were never seen.

While a few previous studies (Guo et al., 2021; Snell et al., 2022) demonstrated the effectiveness of these temporal difference losses, we have empirically observed that in our experiment settings, bootstrapped update leads to relatively unstable learning and poor performance without sensible tuning of hyperparameters (Figure 2).

3.2 Learning Q with expectile regression

Consequently, in this work, we aim to entirely avoid using a bootstrapped update based on a TD error as in $\mathcal{L}_{TD}$. One simple alternative we can think of is the simple Monte-Carlo (MC) estimation of action-values:

$$\min_Q \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T}\left(Q(\mathbf{s}_t,a_t)-G_t\right)^2\right]. \quad (1)$$

Despite its simplicity, it can alleviate some of the weaknesses of $\mathcal{L}_{TD}$ and challenges of the NMT as an offline RL problem: (1) it does not suffer from the problem of querying unseen next action a' during offline training (thus no divergence/less instability), and (2) under the sparse reward function of NMT, Q of any time-step can be directly updated with nonzero reward signal at the end of the trajectory. In this case, we can simply let $G_t = \gamma^{T-t} r_T$.

On the other hand, it is one of the "single-step" methods and will suffer from the same problem that approximated PG suffers from (Figure 1). It is also well known that MC estimation of Q-value has a higher variance compared to TD estimation, which will result in a relatively high variance in gradient and slow learning.

To deal with these problems, we adopt the expectile regression framework and propose the following loss:

$$\mathcal{L}^{\eta}_{ER} = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T} L_2^{\eta}(Q(\mathbf{s}_t,a_t)-G_t)\right], \quad (2)$$

where $L_2^{\eta}(u) = |\eta - \mathbf{1}_{u<0}|u^2$ and $\eta \in (0,1)$. The $Q^{\eta}(\mathbf{s}_t,a_t)$ that minimizes $\mathcal{L}^{\eta}_{ER}$ will be the η-expectile of the return distribution that we can get from $(\mathbf{s}_t,a_t)$, following the behavior policy $\mathcal{D}(a|\mathbf{s})$. Intuitively, it recovers the MC loss (Eq. 1) when $\eta = 0.5$. For $\eta > 0.5$, it assigns smaller weight $1-\eta$ to the return samples G_t that are smaller than $Q^{\eta}(\mathbf{s}_t,a_t)$ while assigning larger weight to the others, resulting in $Q^{\eta}(\mathbf{s}_t,a_t) \geq \mathbb{E}[G_t]$.

Normally this objective would not give a valid Q-value since taking an expectile of the return G_t involves all the stochasticity including those from sampling the transition and reward functions. Fortunately, under the specific MDP formulation that we have made for NMT, we can prove that Q^{η}, a solution to the Eq. (2), is always a valid action value function Q of some specific policy.

Theorem 3.1. *Under the MDP formulation of NMT described in Section 2.1, there exists a policy π such that $Q^{\pi}(s,a) = Q^{\eta}(s,a), \forall (s,a) \in \mathcal{D}$ where Q^{η} is a solution to Eq. (2).*

Proof. See Appendix A.1. □

We also provide another interpretation and a benefit of $\mathcal{L}^{\eta}_{ER}$ by showing what happens when $\eta \to 1$:

Theorem 3.2. *For $Q^{\eta} = \arg\min_Q \mathcal{L}^{\eta}_{ER}$,*

$$\lim_{\eta\to 1} Q^{\eta}(\mathbf{s},a) = \max_{\tau\in\mathcal{D}}\left[\sum_{t=0}^{T}\gamma^t r_t | \mathbf{s}_0=\mathbf{s}, a_0=a\right].$$

Furthermore, the variance of the stochastic gradient $\nabla\tilde{\mathcal{L}}_{ER}$ is zero as we approach the solution,

$$\lim_{\substack{\eta\to 1\\ Q\to Q^{\eta}}} \mathbb{V}[\nabla\tilde{\mathcal{L}}^{\eta}_{ER}] = 0.$$

Proof. See Appendix A.2. □

In other words, as we take $\eta \approx 1$, it is equivalent to letting $Q(\mathbf{s},a)$ be the maximum return we have experienced in the dataset. With high η, the variance of stochastic gradient decreases as we approach the solution, which also alleviates the high variance problem of simple MC Q estimators.

The objective $\mathcal{L}^{\eta}_{ER}$ can be seen as an MC version of IQL (Kostrikov et al., 2021; Snell et al., 2022) to deal with the sparsity of rewards. In general RL problems, this objective may not be desirable since it lacks the ability to "stitch" together different sub-trajectories (Kumar et al., 2020). However, in the RL formulation of NMT, it can be seen that trajectories, which choose different actions a_t^1, a_t^2 at $\mathbf{s}_t$ that give different state transitions $\mathbf{s}_{t+1}^1 \neq \mathbf{s}_{t+1}^2$, will never arrive at the same state in the subsequent time-steps, and no algorithm will benefit from stitching.

Moreover, due to the absence of sampling error, $\mathcal{L}^{\eta}_{ER}$ benefits from a much higher η when compared to Kostrikov et al. (2021). While taking

$\eta \to 1$ is the best choice in principle, such η implies that we completely ignore trajectories other than those that give the max return. Although we observed that in experiments the algorithm does better as we set η higher (Figure 5), there might be cases where smaller η is better due to the worse generalization of the function approximator.

3.3 Conservative expectile regression

We have shown that the $\mathcal{L}^{\eta}_{ER}$ objective introduced in the last section has nice properties, and addresses the problems of offline RL training in principle. However, it should be noted that the analyzed characteristics above apply to the exact Q-function learning, i.e., the tabular case. NMT is at the other extreme where, in the test time, we do not see any of the state-action we have seen during the training and should only rely on the generalization ability of our function approximators.

For example, assume a rare token a that only appears in the dataset one time from some state $\mathbf{s}$, and also assume that it achieves the largest return G in the dataset. Without any other appearance of action a in other states, the function approximator may assume that the return G generalizes to other states, making the translator repeat the same action in any circumstance. It is because the behavior of Q^{η} that approximates the maximum return is only guaranteed for state-actions that appear in the dataset.

Therefore, we adopt the regularization that penalizes uncertain Q-values outside the data support introduced in Conservative Q-learning (Kumar et al., 2020), which optimizes:

$$\mathcal{L}^{\eta}_{CER(\mathcal{H})} = \min_{Q} \mathcal{L}^{\eta}_{ER} + \alpha \mathcal{R}_{\mathcal{H}} \ , \qquad \text{where} \qquad (3)$$

$$\mathcal{R}_{\mathcal{H}} = \mathbb{E}_{\mathcal{D}}\Big[\sum_{t=0}^{T} \log \sum_{\bar{a}} \exp(Q(\mathbf{s}_t, \bar{a})) - Q(\mathbf{s}_t, a_t)\Big]$$

We denote this algorithm as conservative expectile regression: CER($\mathcal{H}$). By adding a regularizer $\mathcal{R}_{\mathcal{H}}$, it will additionally minimize Q-values that lie outside the data support, preventing unseen actions from becoming overly optimistic.

Note that theoretically, the regularization $\mathcal{R}_{\mathcal{H}}$ in CER brings a different amount of conservatism compared to that of conservative Q-learning as we choose to have an asymmetric objective $\mathcal{L}^{\eta}_{ER}$. In the Appendix B, we analyze how much conservatism CER will bring when compared to the oracle-based objective, $\min_Q \mathbb{E}_{\mathcal{D}}[(Q - Q^{\eta})^2] + \alpha\mathcal{R}_{\mathcal{H}}$. The main result is that CER will be at least half conservative when compared to the oracle-based objective when α is small enough. The lower bounds on value functions and objective analyses done by Kumar et al. (2020) can be followed similarly to show that CER provides a valid lower bound on action-value function.

Advantage weighted regularization Interestingly, it can be observed that the regularization term $\mathcal{R}_{\mathcal{H}}$ in Eq. (3) is highly reminiscent of the cross-entropy loss of MLE training if we interpret $Q(\mathbf{s}, \cdot)$ as the *logit* of a token given the current state $\mathbf{s}$, similar to the soft action-value (Guo et al., 2021), i.e. $\tilde{\pi}(a|s) = \texttt{softmax}(Q(\mathbf{s}, a))$. This interpretation naturally lead to a question whether interpolating against the MLE objective is actually beneficial. It will depend on how the dataset is constructed: as more suboptimal trajectories are contained in the dataset, the performance of the policy that MLE objective results alone becomes worse, and so does the optimized policy of CER($\mathcal{H}$). To make CER($\mathcal{H}$) not suffer from the suboptimality of trajectories in the dataset, we need to avoid $\mathcal{R}_{\mathcal{H}}$ over-regularizing towards suboptimal trajectories.

To this end, we also present a variant that regularizes the policy with $\mathcal{R}_{\mathcal{H}}$ but weights the regularization with its importance:

$$\mathcal{R}_{\mathcal{A}} = \mathbb{E}_{\mathcal{D}}\left[-\langle \tilde{\pi}(a|s) \rangle \log \tilde{\pi}(a|\mathbf{s})\right] \text{ with}$$
$$\log \tilde{\pi}(a|\mathbf{s}) = Q(\mathbf{s}, a) - \log \sum_{\bar{a}} \exp(Q(\mathbf{s}, \bar{a})),$$

where $\langle \cdot \rangle$ is a stop-gradient function. We denote this a CER($\mathcal{A}$) algorithm.

3.4 Policy extraction and decoding

As briefly discussed in the last section, the key to better performance in NMT tasks is generalization since the test time state-actions will be never shown to the agent during training. There are two different ways of generalization we can perform with the current algorithm; the generalization in the policy space and the generalization in the action-value space. If we extract a parametrized policy out of the learned action-value function and decode it according to the extracted policy (similar to usual actor-critic algorithms (Kostrikov et al., 2021)), we are generalizing in the policy space. On the other hand, if we directly deploy the learned action-value function for decoding, it corresponds to the generalization in the value function space (Snell et al.,

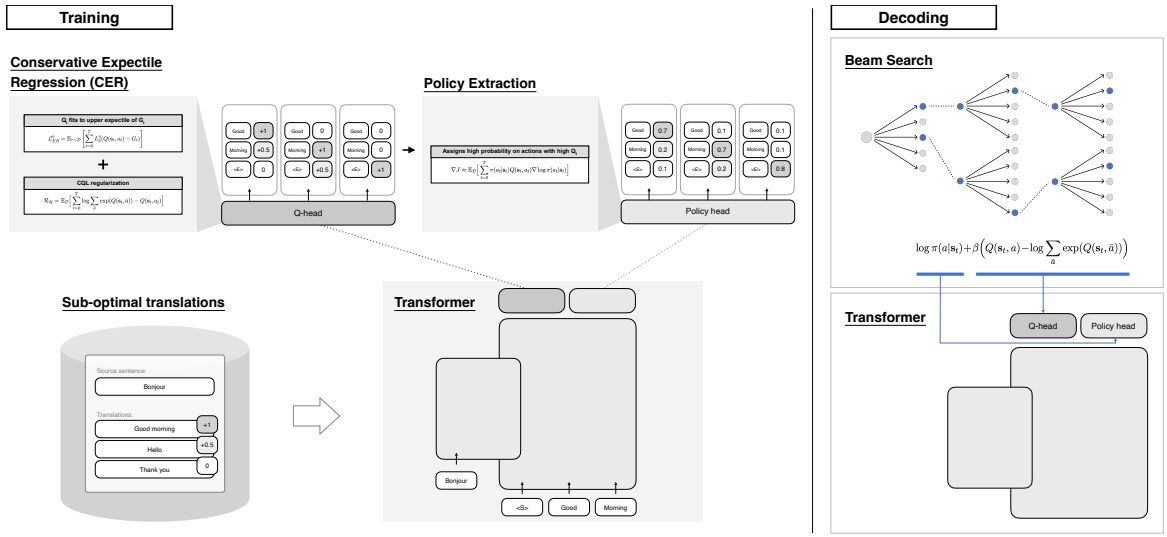

Figure 3: Overall training/decoding algorithms and the architecture of Conservative Expectile Regression. Left: from sub-optimal translations with their evaluations, we train a transformer with two heads: Q-head for an action-value estimation and a policy-head as usual. Q-head is trained with the CER objective (3), and policy extraction is done with off-policy actor-critic style loss (4). Right: we utilize both action-value and policy to perform a beam search during the decoding process.

2022). We empirically found that combining both complements each other and gives the best result.

Policy extraction IQL (Kostrikov et al., 2021) uses a weighted log-likelihood loss with exponentiated advantages to extract a policy, which corresponds to the KL-constrained policy search (Peng et al., 2019). However, as we use additional regularization to penalize OOD actions, we found it unnecessary to constrain it again. We use an off-policy actor-critic style (Degris et al., 2012) policy extraction instead:

$$\nabla J \approx \mathbb{E}_{\mathcal{D}}\Big[\sum_{t=0}^{T} \pi(a_t|\mathbf{s}_t) Q(\mathbf{s}_t, a_t) \nabla \log \pi(a_t|\mathbf{s}_t)\Big] \tag{4}$$

where we assume a uniform data distribution $\mathcal{D}(a|s) = \frac{1}{|\mathcal{D}|}$ similar to approximated PG and hence omitted from above. Sampling actions from $\mathcal{D}$ instead of π enables a far faster policy extraction than by sampling from π. On the other hand, it suffers from a similar issue to the approximated PG, and we add $\min_{s,a} Q(s,a)$ to Q to force positivity.

Decoding We perform a beam search according to the following criterion:

$$\log \pi(a|\mathbf{s}_t) + \beta\Big(Q(\mathbf{s}_t, a) - \log \sum_{\bar{a}} \exp(Q(\mathbf{s}_t, \bar{a}))\Big) \tag{5}$$

to combine the two ways of generalization based on the extracted policy and the learned action-value function, balancing with β. It can be seen as an adaptation of ILQL (Snell et al., 2022) where the second term is designed to mimic an advantage function without explicit estimation of a state value function or the policy.[1]

Another way to interpret this criterion is to see Q as a logit of a policy as we did to develop an advantage weighted regularization. In this case, the criterion corresponds to a weighted sum of two different log probabilities: $\log \pi + \beta \log \tilde{\pi}$.

Architectures for CER The main advantage of CER is that we only need a very minor change in architecture. We use the same transformer architectures that are used for supervised learning of NMT tasks, with an additional head at the top of the decoder to predict an action-value function. Training of the action-value head and the policy extraction can be parallelized as in typical supervised training. Overall training/decoding algorithms and the architecture are summarized in Figure 3.

4 Experiments

Datasets and base architecture In order to benchmark the algorithms for an offline learning with sub-optimal translations, we performed the following procedures. We split the dataset into half, and trained a machine translator using the

[1]Note that the policy π extracted is not a policy that Q is based on, hence $V(s) \neq \mathbb{E}_{\pi}[Q(s,a)]$

	De→En	Ro→En	It→En	Fr→En	En→De	En→Ro	De→En*
Base	31.55	35.05	31.74	39.05	26.05	26.15	29.28
Online PG	33.40	36.64	33.24	39.72	26.77	26.91	30.56
BC	32.56	35.94	32.56	39.6	26.69	27.15	30.15
Approximate PG	32.66	36.35	33.03	39.84	27.14	27.29	30.31
CQL	32.85	36.24	32.91	40.15	27.39	27.60	30.22
ILQL	32.8	36.18	32.97	40.17	27.4	27.56	30.26
CER($\mathcal{H}$)	**33.21**	36.77	33.26	**40.49**	27.59	27.97	30.68
CER($\mathcal{A}$)	33.16	**36.81**	**33.37**	40.47	**27.7**	**27.99**	**30.71**

Table 1: BLEU score comparison for IWSLT'14 and WMT'14 (indicated using *) tasks. Offline algorithms with the highest scores are denoted in bold.

first half of the dataset. We denote this the Base model. Using the Base model, we run beam decoding with a beam size of 50 over the latter half of the dataset, and picked top 5 translations with the best BLEU scores for each source sentences to build an offline RL dataset with sub-optimal translations. SacreBLEU (Post, 2018) is used to evaluate the scores, and the scores are also saved as rewards for translations. These offline datasets thus are 2.5 times larger than the original datasets. We then tested a series of algorithms on these offline RL datasets, warm-starting from the Base model.[2] We constructed offline RL datasets using IWSLT'14 datasets. We also used German-English (De→En) from the WMT'14 translation task for the result on large datasets. See appendix C for more details on experiments.

Methods We implement and compare the following algorithms: behavior cloning (BC), approximate policy gradient (GOLD, Pang and He, 2021), conservative Q learning (CQL, Kumar et al., 2020), implicit language Q-learning (ILQL, Snell et al., 2022) and our algorithm variants, CER($\mathcal{H}$) and CER($\mathcal{A}$). Note that we used the proposed architecture, policy extraction and decoding methods for CQL and ILQL here for the fair comparison, so the only differences are the action-value loss objectives. After the training, the BLEU score is evaluated by performing a beam decoding with a beam size of 5 for all the algorithms.

On the other hand, BC and approximate PG are the policy-based algorithms that do not use additional action-value estimation. BC simply performs supervised training over sub-optimal translations and is independent of the rewards of trajectories. We also report the results of Online PG for the comparison.

Results Table 1 shows the performance of the described experiments, where each entry of a table represents a single run. First, we can note that CQL and ILQL give about the same result, which is obvious considering the sufficiently large η due to the deterministicity of NMT domain. These algorithms improve over approximate PG in most cases, but fails on Ro→En, It→En even with the additional estimation of action values. We believe that this is due to their high sensitivity on regularization coefficient, and the results will improve with more fine-grained hyperparameter search. Nevertheless, it is true that these results make these bootstrapping-based algorithms look less attractive in practice.

On the other hand, our proposed CER algorithms are worth the extra computation for action-value estimation, being consistently better than other competitors. CER($\mathcal{A}$) shows better performance compared to CER($\mathcal{H}$) in most of the cases, although the improvement is marginal. Note that, although CER($\mathcal{A}$) was designed to make the regularization process more intelligent, it is mainly dependent on the accuracy of our action value estimate. CER($\mathcal{A}$) may fail to sufficiently regularize OOD action value estimates if it is large because it less penalizes large action values, and it may result in a smaller gain in practice.

It is hard to compare against online PG as the experiment settings are vastly different, but empirical observations indicate that CER consistently outperforms online PG in a majority of language pairs. Although online PG theoretically has the po-

[2]Note that all the algorithms experimented here have the ability to be trained from scratch; warm-starting was used to reduce training time.

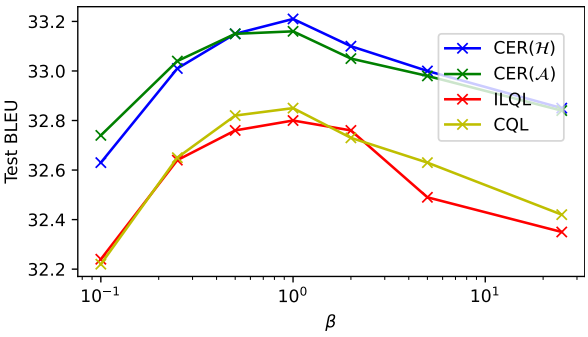

Figure 4: BLEU score on IWSLT'14 De→En of different action-value learning objectives depending on β of beam decoding criterion.

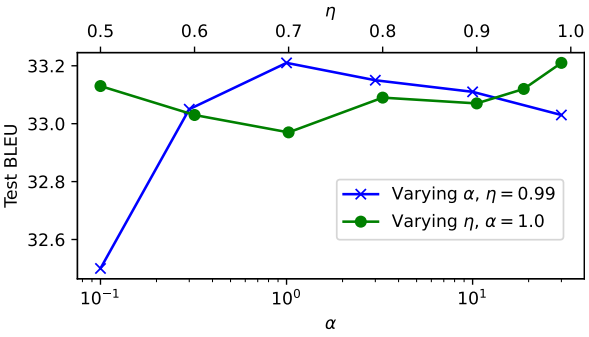

Figure 5: BLEU score on IWSLT'14 De→En of CER($\mathcal{A}$) depending on the choice of hyperparameters α and η.

tential for superior performance in an asymptotic sense, practical inefficiencies arising from online data sampling often hinder its actual achievement of such performance.[3]

4.1 Effect of policy extraction methods

Figure 4 shows the effect of the choice of β of decoding criteron (5) on IWSLT'14 De→En. While $\beta \to 0$ implies we are using the extracted policy only and $\beta \to \infty$ implies we are using the advantage-like perturbation only, we get the best results around $\beta = 1$ regardless of the action-value learning objective. These results show that the policy extraction and decoding methods have large impact on the performance, and that combining the two different generalization methods has a significant benefit in validation performance. In addition, it can be seen that the performance of policy extraction alone is inferior to that of advantage-based decoding alone, suggesting that there is room for improvement in policy extraction.

4.2 Effect of hyperparameters

Figure 5 shows the effect of the choice of hyperparameters α and η of CER($\mathcal{A}$) on IWSLT'14 De→En. Note that we have the following interpretations: $\alpha = 0$ is the expectile regression without conservatism, whereas $\alpha \to \infty$ makes CER the regularization only model, which will be trained like BC (CER($\mathcal{H}$)) or weighted BC (CER($\mathcal{A}$)) where Q is a logit of a token. On the other hand, $\eta \to 1$ corresponds to fitting $Q(\mathbf{s}, a)$ to the maximum return starting from $(\mathbf{s}, a)$, and $\eta \to 0.5$ corresponds to the learning of average return of each state-actions.

[3]Although an increase in BLEU score by < 2 may appear modest, it aligns with previous findings reported, e.g. Kiegeland and Kreutzer (2021).

Results from various alpha experiments show that it is better to use an appropriate α. This is because small α can make translators overly optimistic on high-scoring and rare tokens, while large α reverts the algorithm back to BC. Meanwhile, it is interesting to see how the algorithm performs depending on the different ηs. While it works best with the highest η, the algorithm could also get a high score on the other end $\eta = 0.5$. It can be understood in a way that the effective size of the dataset is the largest when $\eta = 0.5$, and it would have been advantageous for learning better representations. But that leads the algorithm to a "single step" method, and the more sub-optimal translations per source sentence, the worse it will be.

5 Conclusion

In this paper, we develop a novel RL algorithm for NMT, conservative expectile regression, based on expectile regression and a conservative q-learning framework. Based on the unique characteristics of the RL formulation of NMT, the objective of CER is carefully designed to combat all the difficulties we face in the maximization of arbitrary reward signals based on offline RL. We emphasize that CER is as stable as supervised training, and it only needs a slightly more parameters. We have demonstrated the performance by designing an offline RL experiment based on various IWSLT'14 and WMT'14 datasets and shown that CER is clearly advantageous compared to other offline RL algorithms. While these improvement of BLEU scores may not be strongly correlated with human evaluations (Wu et al., 2016), we believe that the experiment aims to demonstrate the efficiency in optimizing the rewards, and the difference is likely to persist even if we optimize for other metrics.

Limitations

While the proposed CER algorithm is a valid algorithm for NMT where the transitions are deterministic, it may not result in an optimal policy when the transitions are stochastic, e.g., dialog management. In this cases, use of ILQL (Snell et al., 2022) is recommended. Furthermore, we were not able to demonstrate the performance of CER algorithm on very large models or on very large amount of data, the impacts of CER on representation learning on those cases are not shown. However, we believe that CER will be at least better than other offline RL algorithms in terms of representation, since it is one of the closest algorithm to supervised training.

Acknowledgements

This work was supported by SAMSUNG Research, Samsung Electronics Co., Ltd. This work was also partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00311, Development of Goal-Oriented Reinforcement Learning Techniques for Contact-Rich Robotic Manipulation of Everyday Objects) and by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2022R1F1A1074880)

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. 2020. On the weaknesses of reinforcement learning for neural machine translation. *ICLR*.

Thomas Degris, Martha White, and Richard Sutton. 2012. Off-policy actor-critic. In *International Conference on Machine Learning*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, New Orleans, Louisiana. Association for Computational Linguistics.

Han Guo, Bowen Tan, Zhengzhong Liu, Eric P Xing, and Zhiting Hu. 2021. Text generation with efficient (soft) q-learning. *arXiv preprint arXiv:2106.07704*.

Samuel Kiegeland and Julia Kreutzer. 2021. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1673–1681, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. *ICLR*.

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. 2021. Dr3: Value-based deep reinforcement learning requires explicit regularization. In *ICLR*.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. 2021. What matters in learning from offline human demonstrations for robot manipulation. *CoRL*.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1464–1474, Copenhagen, Denmark. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Richard Yuanzhe Pang and He He. 2021. Text generation by learning from demonstrations. *ICLR*.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. 2022. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

A Proofs

We provide the proofs for the theorems here.

A.1 Proof of Theorem 3.1

Lemma A.1. *Assume two sets of real numbers* $X_1 = \{x_1, ..., x_n\}$ *and* $X_2 = \{x_{n+1}, ..., x_{n+m}\}$ *with* $x_i \in \mathbb{R}\ \forall i$. *Define an* η*-expectile of a set:*

$$m_\eta(X) = \arg\min_{m_\eta} \frac{1}{|X|} \sum_{x \in X} L_2^\eta(x - m_\eta). \quad (6)$$

Then, it satisfies $\min(m_\eta(X_1), m_\eta(X_2)) \leq m_\eta(X_1 \cup X_2) \leq \max(m_\eta(X_1), m_\eta(X_2))$.

Proof. By the first order condition of $m_\eta(X)$, it satisfies

$$\eta \sum_{x \in X} (x - m_\eta)^+ = (\eta - 1) \sum_{x \in X} (x - m_\eta)^-. \quad (7)$$

W.l.o.g. assume $m_\eta(X_1) \leq m_\eta(X_2)$. Assume that $m_\eta(X_1 \cup X_2) < \min(m_\eta(X_1), m_\eta(X_2)) = m_\eta(X_1)$. It means

$$\eta \sum_{X_1 \cup X_2} (x - m_\eta(X_1))^+ < (\eta - 1) \sum_{X_1 \cup X_2} (x - m_\eta(X_1))^-. \quad (8)$$

Subtracting Eq. (7) from Eq. (8) implies that $m_\eta(X_2) < m_\eta(X_1)$, which is a contradiction. Therefore, $m_\eta(X_1 \cup X_2) \geq m_\eta(X_1) = \min(m_\eta(X_1), m_\eta(X_2))$. The other way around can be proved similarly. □

Theorem A.1. *Under the MDP formulation of NMT described in Section 2.1, there exists a policy* π *such that* $Q^\pi(s, a) = Q^\eta(s, a), \forall (s, a) \in \mathcal{D}$ *where* Q^η *is a solution to Eq.* (2).

Proof. We use the following properties of the MDP formulation of NMT: (1) the transition function is deterministic, and (2) the same state is never visited more than once.

We prove this by induction. From the terminal states $\mathbf{s}$, we have a set of actions that have tried in $\mathbf{s}$: $\{a_i\}_i$. Since the same state is never visited more than once, we have $Q^\eta(\mathbf{s}, a_i) = Q^\pi(\mathbf{s}, a_i) = R(\mathbf{s}, a_i)\ \forall i$ for any η, π.

Now pick an arbitrary non-terminal state-action pair from the dataset $(\mathbf{s}, a) \in \mathcal{D}$. It will give a deterministic next state $\mathbf{s}'$. There will be the next actions $\{a'_i\}_i$ that have been chosen from $\mathbf{s}'$. Assume that there exists π such that $Q^\pi(\mathbf{s}', a'_i) = Q^\eta(\mathbf{s}', a'_i)\ \forall i$

for some fixed η. These $Q^\eta(\mathbf{s}', a'_i)$, which is the solution to Eq. (2), are expectiles of sets of returns starting from $(\mathbf{s}', a'_i)$ and they are equal to $Q^\pi(\mathbf{s}', a'_i)$ from the assumption, i.e.,

$$Q^\pi(\mathbf{s}', a'_i) = m_\eta(\{G_t : G_t \in \mathcal{D} | \mathbf{s}_t = \mathbf{s}', a_t = a'_i\}).$$

Note that according to Eq. (7), if we shift and scale the data by constants, the expectile will also be shifted and scaled by the same amount. Therefore, $R(\mathbf{s}, a) + \gamma Q^\pi(\mathbf{s}', a'_i)$ are also expectiles of $R(\mathbf{s}, a) + \gamma G_t$s with G_ts starting from $(\mathbf{s}', a'_i)$.

$$R(\mathbf{s}, a) + \gamma Q^\pi(\mathbf{s}', a'_i) = \\ m_\eta(\{R(\mathbf{s}, a) + \gamma G_t : G_t \in \mathcal{D} | \mathbf{s}_t = \mathbf{s}', a_t = a'_i\}).$$

Now observe that $Q^\eta(\mathbf{s}, a)$ is an expectile of G_{t-1}s starting from $(\mathbf{s}, a)$, which is a union of $R(\mathbf{s}, a) + \gamma G_t$s starting from $\{(\mathbf{s}', a'_i)\}_i$s:

$$\begin{aligned} &Q^\eta(\mathbf{s}, a) \\ &= m_\eta(\{G_{t-1} : G_{t-1} \in \mathcal{D} | \mathbf{s}_{t-1} = \mathbf{s}, a_{t-1} = a\}) \\ &= m_\eta\Big(\bigcup_i \{R(\mathbf{s}, a) + \gamma G_t : G_t \in \mathcal{D} | \mathbf{s}_t = \mathbf{s}', a_t = a'_i\}\Big). \end{aligned}$$

Using Lemma A.1, we see that

$$\begin{aligned} &\min_i R(\mathbf{s}, a) + \gamma Q^\pi(\mathbf{s}', a'_i) \\ &\quad \le Q^\eta(\mathbf{s}, a) \le \max_i R(\mathbf{s}, a) + \gamma Q^\pi(\mathbf{s}', a'_i). \end{aligned}$$

This implies that there exists a convex combination $0 \le \lambda_i \le 1, \sum_i \lambda_i = 1$ such that

$$\begin{aligned} Q^\eta(\mathbf{s}, a) &= \sum_i \lambda_i \left(R(\mathbf{s}, a) + \gamma Q^\pi(\mathbf{s}', a'_i)\right) \\ &= R(\mathbf{s}, a) + \gamma \sum_i \lambda_i Q^\pi(\mathbf{s}', a'_i). \end{aligned}$$

Therefore, by choosing $\pi(a_i|\mathbf{s}) = \lambda_i \ \forall i$, we see that $Q^\eta(\mathbf{s}, a) = Q^\pi(\mathbf{s}, a)$. Furthermore, choosing π accordingly does not affect Q^π after $(\mathbf{s}, a)$ since we do not visit same state more than once. □

A.2 Proof of Theorem 3.2

Theorem A.2. *For $Q^\eta = \arg\min_Q \mathcal{L}^\eta_{ER}$,*

$$\lim_{\eta \to 1} Q^\eta(\mathbf{s}, a) = \max_{\tau \in \mathcal{D}} \left[\sum_{t=0}^{T} \gamma^t r_t | \mathbf{s}_0 = \mathbf{s}, a_0 = a\right].$$

Furthermore, the variance of the stochastic gradient $\nabla \tilde{\mathcal{L}}_{ER}$ is zero as we approach the solution,

$$\lim_{\substack{\eta \to 1 \\ Q \to Q^\eta}} \mathbb{V}[\nabla \tilde{\mathcal{L}}^\eta_{ER}] = 0.$$

Proof. The proof of the first statement mainly follow the proof of Lemma 1 of (Kostrikov et al., 2021). As in the proof of Lemma A.1, first order condition of $Q^\eta(\mathbf{s}, a)$ is:

$$\eta \sum_{G \in \mathcal{G}} (G - Q^\eta(\mathbf{s}, a))^+ = (\eta - 1) \sum_{G \in \mathcal{G}} (G - Q^\eta(\mathbf{s}, a))^-.$$

where $\mathcal{G} = \{G_t : G_t \in \mathcal{D} | \mathbf{s}_t = \mathbf{s}, a_t = a\}$. If $Q^\eta(\mathbf{s}, a) > \max_{G \in \mathcal{G}} G$, the above condition cannot be satisfied since LHS is 0, and RHS is larger than 0. This implies $Q^\eta(\mathbf{s}, a) \le \max_{G \in \mathcal{G}} G$ for any η.

Moreover, assume η_1 and η_2 with $\eta_1 < \eta_2$. It can be easily seen that

$$\begin{aligned} &\eta_2 \sum_{G \in \mathcal{G}} (G - Q^{\eta_1}(\mathbf{s}, a))^+ \\ &\qquad \ge (\eta_2 - 1) \sum_{G \in \mathcal{G}} (G - Q^{\eta_1}(\mathbf{s}, a))^-, \end{aligned}$$

and the gap will only narrow when $Q(\mathbf{s}, a)$ increases. This means $Q^{\eta_1}(\mathbf{s}, a) \ge Q^{\eta_2(\mathbf{s}, a)}$ where the equality only holds for the trivial case when all $G \in \mathcal{G}$ are the same. Then η is bounded and monotonically increasing function except the trivial case, and the limit $\lim_{\eta \to 1} Q^\eta(\mathbf{s}, a) = \max_{G \in \mathcal{G}} G$ follows.

For the second statement, we redefine the loss to derive the following stochastic gradient:

$$\begin{aligned} \mathcal{L}^\eta_{ER} &= \mathbb{E}_{(\mathbf{s}, a, G) \sim \mathcal{D}}[\tilde{\mathcal{L}}^\eta_{ER}(\mathbf{s}, a, G)] \\ \tilde{\mathcal{L}}^\eta_{ER}(\mathbf{s}, a, G) &= |\eta - \mathbf{1}_{Q(\mathbf{s}, a) < G}| (Q(\mathbf{s}, a) - G)^2 \\ \nabla \tilde{\mathcal{L}}^\eta_{ER}(\mathbf{s}, a, G) &= 2|\eta - \mathbf{1}_{Q(\mathbf{s}, a) < G}| (Q(\mathbf{s}, a) - G) \\ &= -2\eta (G - Q(\mathbf{s}, a))^+ \\ &\quad - 2(1 - \eta)(G - Q(\mathbf{s}, a))^- \end{aligned}$$

for $Q(\mathbf{s}, a) \ne G$. Then,

$$\begin{aligned} &\lim_{\substack{\eta \to 1 \\ Q \to Q^\eta}} (G - Q(\mathbf{s}, a))^+ = (G - \max_{G \in \mathcal{G}} G)^+ = 0 \\ &\lim_{\eta \to 1} (1 - \eta)(G - Q(\mathbf{s}, a))^- = 0 \end{aligned}$$

and $\lim_{\substack{\eta \to 1 \\ Q \to Q^\eta}} \mathbb{V}[\nabla \tilde{\mathcal{L}}^\eta_{ER}] = 0.$ □

B Analysis on Conservative Expectile Regression

Recall the objective of CER that we introduced:

$$\begin{aligned} &\mathcal{L}^\eta_{ER} + \alpha \mathcal{R}_{CQL} \\ &= \mathbb{E}_{\mathcal{D}}[|\eta - \mathbf{1}_{Q(\mathbf{s}, a) < G}| (Q(\mathbf{s}, a) - G)^2] \\ &+ \alpha \mathbb{E}_{\mathcal{D}}[\mathbb{E}_{\bar{a} \sim \mu(a|\mathbf{s})} Q(\mathbf{s}, \bar{a}) - Q(\mathbf{s}, a)]. \end{aligned}$$

Let the derivative of the above objective to be 0:

$$
\begin{aligned}
0 &= \nabla(\mathcal{L}^{\eta}_{ER} + \alpha\mathcal{R}_{CQL}) \\
&= \mathbb{E}_{\mathcal{D}}\Big[-2\eta(G - Q(\mathbf{s},a))^{+} \\
&\quad - 2(1-\eta)(G - Q(\mathbf{s},a))^{-} + \alpha\left(\frac{\mu(a|\mathbf{s})}{\mathcal{D}(a|\mathbf{s})} - 1\right)\Big]
\end{aligned}
$$

Fix $\mathbf{s}, a \in \mathcal{D}$, and define the return distribution $p_G(G|\mathbf{s},a)$ that can be computed from $\mathcal{D}$. Equivalently, we can write:

$$
\begin{aligned}
\frac{\alpha}{2}\left(\frac{\mu(a|\mathbf{s})}{\mathcal{D}(a|\mathbf{s})} - 1\right) &= \mathbb{E}_{p_G}\Big[\eta(G - Q(\mathbf{s},a))^{+} \\
&\quad + (1-\eta)(G - Q(\mathbf{s},a))^{-}\Big] \\
&= \mathbb{E}_{p_G}\Big[(2\eta - 1)(G - Q(\mathbf{s},a))^{+}\Big] \\
&\quad + (1-\eta)(\mathbb{E}_{p_G}[G] - Q(\mathbf{s},a))
\end{aligned}
$$

and we get:

$$
\begin{aligned}
Q(\mathbf{s},a) = \mathbb{E}_{p_G}\Big[G + \frac{(2\eta-1)}{(1-\eta)}(G - Q(\mathbf{s},a))^{+}\Big] \\
- \frac{\alpha}{2(1-\eta)}\left(\frac{\mu(a|\mathbf{s})}{\mathcal{D}(a|\mathbf{s})} - 1\right).
\end{aligned}
$$

Now, we define the portion of return samples G that is larger than Q to be $\rho_{G>Q} \in [0,1]$, and the average of Gs that is larger than Q to be $G_{>Q}$. Then, we can write:

$$
\mathbb{E}_{p_G}[(G - Q(\mathbf{s},a))^{+}] = \mathbb{E}_{p_G}[\rho_{G>Q}(G - Q(\mathbf{s},a))]
$$

These two newly defined $\rho_{G>Q}, G_{>Q}$ depend on Q, but they do not change until Q moves and passes another G sample. Based on those, we can derive the following:

$$
\begin{aligned}
\left(1 - \rho_{G>Q}\frac{(2\eta-1)}{(1-\eta)}\right)Q(\mathbf{s},a) \\
= \mathbb{E}_{p_G}\left[G + \frac{(2\eta-1)}{(1-\eta)}\rho_{G>Q}G_{>Q}\right] \\
- \frac{\alpha}{2(1-\eta)}\left(\frac{\mu(a|\mathbf{s})}{\mathcal{D}(a|\mathbf{s})} - 1\right),
\end{aligned}
$$

arranging the terms we get,

$$
\begin{aligned}
Q(\mathbf{s},a) = c_1\mathbb{E}_{p_G}[G] + c_2\mathbb{E}_{p_G}[G_{>Q}] \\
- c_3\alpha\left(\frac{\mu(a|\mathbf{s})}{\mathcal{D}(a|\mathbf{s})} - 1\right)
\end{aligned}
$$

where $c_1 = \frac{1-\eta}{1-\eta-\rho_{G>Q}(2\eta-1)}$, $c_2 = \frac{\rho_{G>Q}(2\eta-1)}{1-\eta-\rho_{G>Q}(2\eta-1)}$, and $c_3 = \frac{1}{2(1-\eta-\rho_{G>Q}(2\eta-1))}$.

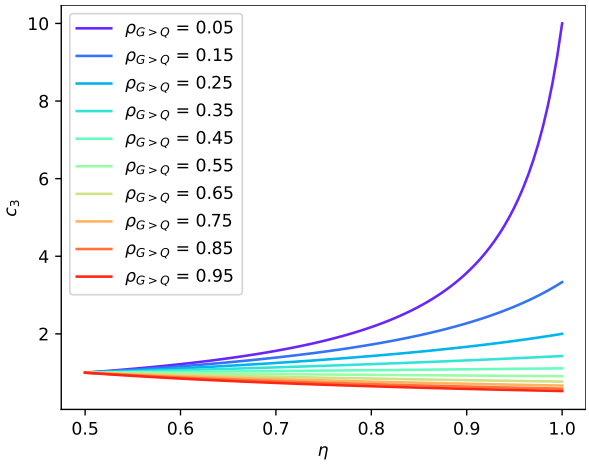

Figure 6: The relationship between η, c_3, and $\rho_{G>Q}$.

If we know the expectile $Q^{\eta}(\mathbf{s},a)$ in advance, the objective that resemble the original conservative Q-learning the most would be the loss based on symmetric least squares:

$$
\mathbb{E}_{\mathcal{D}}\left[\frac{1}{2}(Q(\mathbf{s},a) - Q^{\eta}(\mathbf{s},a))^2\right] + \alpha_o\mathcal{R}_{CQL}.
$$

With similar derivations with new oracle-based objective, we get:

$$
\begin{aligned}
Q(\mathbf{s},a) &= Q^{\eta}(\mathbf{s},a) - \alpha_o\left(\frac{\mu(a|\mathbf{s})}{\mathcal{D}(a|\mathbf{s})} - 1\right) \\
&= c_1\mathbb{E}_{p_G}[G] + c_2\mathbb{E}_{p_G}[G_{>Q}] \\
&\quad - \alpha_o\left(\frac{\mu(a|\mathbf{s})}{\mathcal{D}(a|\mathbf{s})} - 1\right)
\end{aligned}
$$

Therefore, given that $\rho_{G>Q}, G_{>Q}$ do not change (i.e., change on $Q(\mathbf{s},a)$ due to $\mathcal{R}_{CQL}$ is not big enough to lower $Q(\mathbf{s},a)$ to pass some G), the solution to the CER algorithm is equivalent to that of oracle-based objective by letting $\alpha_o = c_3\alpha$. Looking into c_3, we can see that $\min_{\eta,\rho_{G>Q}} c_3 = \frac{1}{2}$, showing that CER will be at least half conservative when compared to oracle-based objective.

In Figure 6, we show the relationship between η, c_3, and $\rho_{G>Q}$. It can be seen that $c_3 \to 1$ as $\eta \to 0.5$, which is well aligned with our discussion since $\eta = 0.5$ makes $\mathcal{L}^{\eta}_{ER}$ to be a MSE loss and CER is equivalent to the oracle-based objective. On the other hand, when $\eta \to 1$, the amount of conservativeness varies much depending on how the data is distributed: $\rho_{G>Q}$. While c_3 may grow to arbitrarily large value in the extreme case of $\rho_{G>Q} \to 0.0, \eta \to 1.0$, such a small $\rho_{G>Q}$ can hardly be achieved in practice, where we have only few return samples per same state-action (at most 5 in our experiments).

C Experiment details

We provide all the experiment details that are not provided in the main text here. Models are optimized with Adam (Kingma and Ba, 2015) with parameters $\beta = 0.9$ and $\beta_2 = 0.98$. Training starts with linear warmup for 4×10^3 steps until it reaches the learning rate 10^{-4}, and then inverse square root learning rate scheduler is used to schedule the learning rate. 0.3 dropout probability and weight decay of coefficient 10^{-4} are used. Base models are trained over 40 epochs, where the offline training is done over 20 epochs. We used label smoothing (0.1) for the supervised training of Base model. Note that the amount of data is different between these two. We ran all experiments on 4 Nvidia GTX 3090 GPU, and running all experiments took about 500 GPU hours.

Dataset and Architectures We constructed offline RL datasets using German-English (De→En), Romanian-English (Ro→En), Italian-English (It→En) and French-English (Fr→En) from IWSLT'14 datasets. We also experimented on IWSLT'14 English-German (En→De) and English-Romanian (En→Ro) to see the results on different target languages. We used byte-pair-encoding (Sennrich et al., 2016) to preprocess all sentences. For these tasks, tst2010, tst2011, tst2012 datasets are merged to form test datasets and report BLEU on these datasets. We also used German-English (De→En) from the WMT'14 translation task for the result on large datasets.

We use the Fairseq (Ott et al., 2019) implementation of the Transformers architecture (Vaswani et al., 2017). We used the transformer architecture of six encoder layers, six decoder layers, 4 attention heads, 512 embedding dimension and 1024 inner-layer dimension for all the IWSLT'14 experiments.[4] For WMT'14 experiments, we used 8 attention heads and 2048 inner-layer dimension instead.[5]

Hyperparameters For approximate PG, we used truncated approximated importance weight $w_t = \max(\pi(a_t|\mathbf{s}_t), 0.1)$, following Pang and He (2021). For regularization coefficients α for CQL and ILQL, we reported the results with $\alpha = 10^2$, which resulted in the best performance out of the set of $\{10^0, 10^1, 10^2\}$. For the expectile hyperparameter of ILQL, we used $\eta = 0.99$ that resulted in the best performance out of the set of $\{0.9, 0.95, 0.99\}$. For CERs, $\alpha = 10^0$ and $\eta = 0.99$ is used.

[4] `transformer_iwslt_de_en` architecture of Fairseq.

[5] `transformer_wmt_en_de` architecture of Fairseq.