Reinforcement Learning based Curriculum Optimization for Neural Machine Translation

Gaurav Kumar
Johns Hopkins University
gkumar@cs.jhu.edu

George Foster, Colin Cherry, Maxim Krikun
Google AI
{fosterg, colincherry, krikun}
@google.com

Abstract

We consider the problem of making efficient use of heterogeneous training data in neural machine translation (NMT). Specifically, given a training dataset with a sentence-level feature such as noise, we seek an optimal *curriculum*, or order for presenting examples to the system during training. Our curriculum framework allows examples to appear an arbitrary number of times, and thus generalizes data weighting, filtering, and fine-tuning schemes. Rather than relying on prior knowledge to design a curriculum, we use reinforcement learning to learn one automatically, jointly with the NMT system, in the course of a single training run. We show that this approach can beat uniform baselines on Paracrawl and WMT English-to-French datasets by +3.4 and +1.3 BLEU respectively. Additionally, we match the performance of strong filtering baselines and hand-designed, state-of-the-art curricula.

1 Introduction

Machine Translation training data is typically heterogeneous: it may vary in characteristics such as domain, translation quality, and degree of difficulty. Many approaches have been proposed to cope with heterogeneity, such as filtering (Duh et al., 2013) or down-weighting (Wang et al., 2017) examples that are likely to be noisy or out of domain. A powerful technique is to control the curriculum—the order in which examples are presented to the system—as is done in fine-tuning (Freitag and Al-Onaizan, 2016), where training occurs first on general data, and then on more valuable in-domain data. Curriculum based approaches generalize data filtering and weighting[1] by allowing examples to be visited multiple times or not at all; and they additionally potentially enable steering the training trajectory toward a better global optimum than might be attainable with a static attribute-weighting scheme.

Devising a good curriculum is a challenging task that is typically carried out manually using prior knowledge of the data and its attributes. Although powerful heuristics like fine-tuning are helpful, setting hyper-parameters to specify a curriculum is usually a matter of extensive trial and error. Automating this process with meta-learning is thus an attractive proposition. However, it comes with many potential pitfalls such as failing to match a human-designed curriculum, or significantly increasing training time.

In this paper we present an initial study on meta-learning an NMT curriculum. Starting from scratch, we attempt to match the performance of a successful non-trivial *reference curriculum* proposed by Wang et al. (2018), in which training gradually focuses on increasingly cleaner data, as measured by an external scoring function. Inspired by Wu et al. (2018), we use a reinforcement-learning (RL) approach involving a learned agent whose task is to choose a corpus bin, representing a given noise level, at each NMT training step. A challenging aspect of this task is that choosing only the cleanest bin is sub-optimal; the reference curriculum uses all the data in the early stages of training, and only gradually anneals toward the cleanest. Furthermore, we impose the condition that the agent must learn its curriculum in the course of a single NMT training run.

We demonstrate that our RL agent can learn a curriculum that works as well as the reference, obtaining a similar quality improvement over a random-curriculum baseline. Interestingly, it does so using a different strategy from the reference. This result opens the door to learning more sophisticated curricula that exploit multiple data at-

[1] Assuming integer weights.

Proceedings of NAACL-HLT 2019, pages 2054–2061
Minneapolis, Minnesota, June 2 - June 7, 2019.

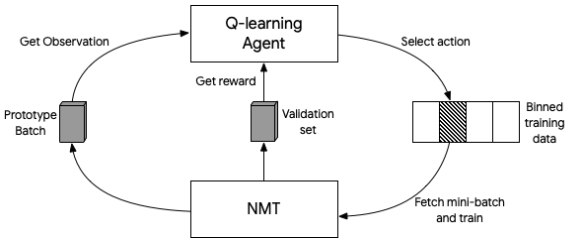


Figure 1: The agent's interface with the NMT system.

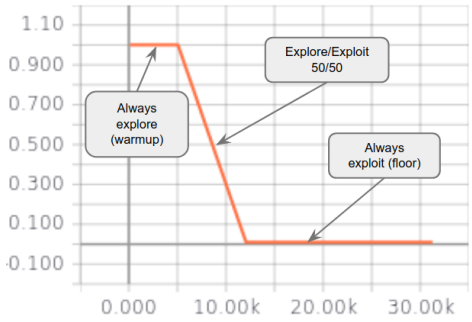


Figure 2: Linearly-decaying ϵ-greedy exploration.

tributes and work with arbitrary corpora.

2 Related Work

Among the very extensive work on handling heterogeneous data in NMT, the closest to ours are techniques that re-weight (Chen et al., 2017) or re-order examples to deal with domain mismatch (van der Wees et al., 2017; Sajjad et al., 2017) or noise (Wang et al., 2018).

The idea of a curriculum was popularized by Bengio et al. (2009), who viewed it as a way to improve convergence by presenting heuristically-identified easy examples first. Several recent papers (Kocmi and Bojar, 2017; Zhang et al., 2019; Platanios et al., 2019) explore similar ideas for NMT, and verify that this strategy can reduce training time and improve quality.

Work on meta-learning a curriculum originated with Tsvetkov et al. (2016), who used Bayesian optimization to learn a linear model for ranking examples in a word-embedding task. This approach requires a large number of complete training runs, and is thus impractical for NMT. More recent work has explored bandit optimization for scheduling tasks in a multi-task problem (Graves et al., 2017), and reinforcement learning for selecting examples in a co-trained classifier (Wu et al., 2018). Finally, Liu et al. (2018) apply imitation learning to actively select monolingual training sentences for labeling in NMT, and show that the learned strategy can be transferred to a related language pair.

3 Methods

The attribute we choose to learn a curriculum over is noise. To determine a per-sentence noise score, we use the *contrastive data selection* (CDS) method defined in Wang et al. (2018). Given the parameters θ_n of an NMT model trained on a noisy corpus, and parameters θ_c of the same model fine-tuned on a very small trusted corpus, the score $s(e, f)$ for a translation pair e, f is defined as:

$$s(e, f) = \log p_{\theta_c}(f|e) - \log p_{\theta_n}(f|e) \quad (1)$$

Wang et al. (2018) show that this correlates very well with human judgments of data quality. They use the CDS score in a heuristic, online schedule that slowly anneals from sampling mini-batches from all the training data to sampling only from the highest-scoring (cleanest) data. Our goal is to replace this heuristic curriculum with a learned one.

Q-learning for NMT Curricula

Our agent uses deep Q-learning (DQN) (Mnih et al., 2015) which is a model-free reinforcement learning procedure. The agent receives an *observation* from the environment and conditions on it to produce an *action* which is executed upon the environment. It then receives a *reward* representing the goodness of the executed action. The agent chooses actions according to a state-action value (Q) function, and attempts to learn the Q-function so as to maximize expected total rewards.

In our setup, the environment is the NMT system and its training data, as illustrated in Figure 1. We divide the training data into a small number of equal-sized *bins* according to CDS scores. At each step, the agent selects a bin (action) from which a mini-batch is sampled to train the NMT system.

Our RL agent must balance exploration (choosing an action at random) versus exploitation (choosing the action which maximizes the Q-function). In our setup, this is done using a linearly-decaying ϵ-greedy exploration strategy (Figure 2). This strategy has three phases: (1) The warmup period where we always explore; (2) the decay period where the probability of exploration decreases and exploitation increases; (3) the floor where we almost always exploit. Since we

do not want to exploit an uninformed Q-function, the duration of exploration needs to be set carefully. In our experiments, we found that longer decays were useful and the best performance was achieved when the decay was set to about 50% of the expected NMT training steps.

Observation Engineering

The *observation* is meant to be a summary of the state of the environment. The NMT parameters are too numerous to use as a sensible observation at each time step. Inspired by Wu et al. (2018), we propose an observation type which is a function of the NMT system's current performance at various levels of noise. We first create a *prototype batch* by sampling a fixed number of prototypical sentences from each bin of the training data. At each time step, the observation is the vector containing sentence-level log-likelihoods produced by the NMT system for this prototype batch.

Since the observations are based on likelihood, a metric which aggressively decays at the beginning of NMT training, we use an NMT warmup period to exclude this period from RL training. Otherwise, the initial observations would be unlike any that occur later.

Reward Engineering

Our objective is to find a curriculum which maximizes the likelihood of the NMT system on a development set. The RL *reward* that directly corresponds to this goal would be the highest likelihood value reached during an NMT training run. However, as we use only one NMT training run, having a single reward per run is infeasible. To provide a denser signal to the RL agent, we define the reward at a step to be the change in likelihood since the most recent previous step for which development-set likelihood is available. This has the desired property that the sum of per-step rewards maximized by the RL agent is equal to the NMT maximum-likelihood objective (on development data). We rely on the WMT warmup period described in the previous section to eliminate spuriously large rewards at the beginning of training.

4 Experimental Setup

Our NMT model is similar to RNMT+ (Chen et al., 2018), but with only four layers in both encoder and decoder. Rewards (dev-set log-likelihood) are provided approximately every 10 training steps by an asynchronous process.

We use the DQN agent implementation in Dopamine,[2] which includes an experience replay buffer to remove temporal correlations from the observations, among other DQN best practices. Due to the sparse and asynchronous nature of our rewards, we store observation, action transitions in a temporary buffer until a new reward arrives. At this point, transitions are moved from the temporary buffer to the DQN agent's replay buffer. The RL agent is trained after each NMT training step by sampling an RL mini-batch from the replay buffer. Our RL hyper-parameter settings are listed in the appendix.

Following Wang et al. (2018), we use the Paracrawl and WMT English-French corpora for our experiments. These contain 290M and 36M training sentences respectively. WMT is relatively clean, while a large majority of Paracrawl sentence pairs contain noise. We process both corpora with BPE, using a vocabulary size of 32k. Both corpora are split into 6 equal-sized bins according to their noise level, as provided by CDS score. In both settings, the WMT newstest 2010-2011 corpus is used as trusted data for CDS scores, which are computed using the models and procedure described in Wang et al. (2018). For the *prototype batch* used to generate observations, we extracted the 32 sentences whose CDS scores are closest to the mean in each bin, giving a total of 192 sentences. We use WMT 2012-2013 for development and WMT 2014 for test, and report tokenized, naturally-cased BLEU scores from the test checkpoint closest to the highest-BLEU dev checkpoint. To combat variance caused by sampling different batches per bin (which produces somewhat different results even when bins are visited in fixed order), all models were run twice with different random seeds, and the model with the best score on the dev set was chosen.

5 Results

Our results are presented in Table 1. **Uniform baselines** consist of:

- *Uniform* – standard NMT training
- *Uniform (6-bins)* – sample a bin uniformly at random, and then sample a mini-batch from that bin

[2] github.com/google/dopamine

	Paracrawl	WMT
Uniform baselines		
Uniform	34.1	37.1
Uniform (6-bins)	34.8	-
Uniform (bookends)	35.0	34.8
Heuristic baselines		
Filtered (20%/33%)	37.0	38.3
Fixed ϵ-schedule	36.9	37.7
Online	37.5	37.7
Learned curricula		
Q-learning (bookends)	36.8	36.3
Q-learning (6-bins)	37.5	38.4

Table 1: BLEU scores on Paracrawl and WMT En-Fr datasets with uniform, heuristic and learned curricula.

- *Uniform (bookends)* – as Uniform (6-bins) but uniformly sampling over just the best and worst bin.

Surprisingly, 6-bins performs better than the standard NMT baseline. We hypothesize that this can be attributed to more homogeneous mini-batches.

Heuristic baselines are:

- *Filtered* – train only on the highest-quality data as determined by CDS scores: top 20% of the data for Paracrawl, top 33% for WMT.
- *Fixed ϵ-schedule* – we use the ϵ-decay strategy of our best RL experiment, but always choose the cleanest bin when we exploit.
- *Online* – the online schedule from Wang et al. (2018) adapted to the 6-bin setting. We verified experimentally that our performance matched the original schedule, which did not use hard binning.

Learned curricula were trained over 2 bookend (worst and best) bins and all 6 bins. On the Paracrawl dataset, in the 2-bin setting, the learned curriculum beats all uniform baselines and almost matches the optimized filtering baseline.[3] With 6-bins, it beats all uniform baselines by up to 2.5 BLEU and matches the hand-designed online baseline of Wang et al. (2018). On WMT, with 2 bins, the learned curriculum beats the 2-bin baseline, but not the uniform baseline over all data.

[3]The clean data available in the 2-bin setup is limited to the best bin (16%), while filtering uses slightly more data (20%).

Reward \ Observation	Default	Fixed
Default	**37.5**	37.5
Fixed	32.5	-

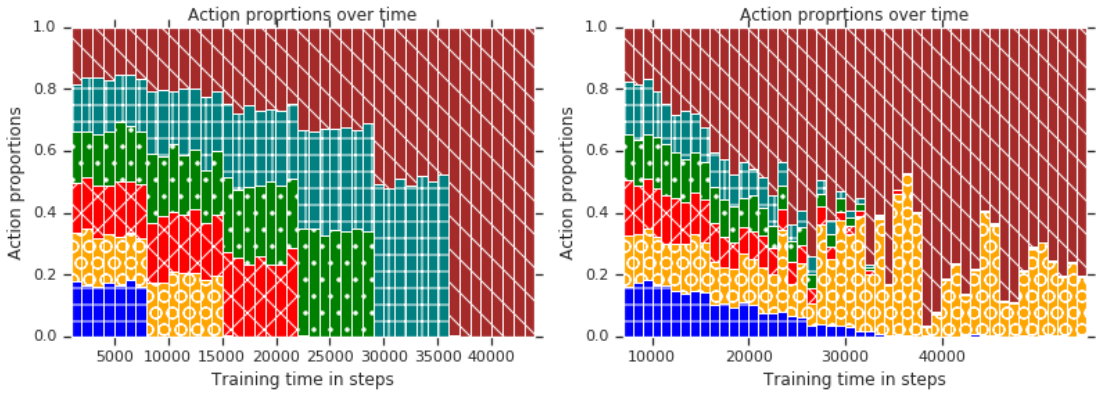
Table 2: BLEU scores on ablation experiments with fixed rewards or observations on the Paracrawl En-Fr dataset.

With 6 bins, the learned curriculum beats the uniform baseline by 1.5 BLEU, and matches the filtered baseline, which in this case outperforms the online curriculum by 0.6 BLEU.

Our exploration strategy for Q-learning (see Figure 2) forces the agent to visit all bins during initial training, and only gradually rely on its learned policy. This mimics the gradual annealing of the online curriculum, so one possibility is that the agent is simply choosing the cleanest bin whenever it can, and its good performance comes from the enforced period of exploration. However, the fact that the agent beats the fixed ϵ-schedule (see Table 1) described above on both corpora makes this unlikely.

6 Analysis

Task-specific reward and observation engineering is critical when building an RL model. We performed ablation experiments to determine if the rewards and observations we have chosen contain information which aids us in the curriculum learning task. Table 2 shows the results of our experiments. The fixed reward experiments were conducted by replacing the default delta-perplexity based reward with a static reward which returns a reward of one when the cleanest bin was selected and zero otherwise. The fixed observation experiments used a static vector of zeroes as input at all time steps. Using fixed observations matches the performance of dynamic observations, from which we can draw two conclusions. First, the agent's good performance is due to associating higher rewards with better bins, but it learns to do so slowly (partly modulated by its ϵ-greedy schedule) so that it avoids the sub-optimal strategy of choosing only the best bin. Second, its ability to distinguish among bins is not impeded by the use of an observation vector that slowly evolves through time and never returns to previous states.



(a) Online

(b) RL learned

Figure 3: Online policy from Wang et al. (2018) compared to the RL policy. Each color/pattern represents a bin (blue is the noisiest bin, dark red is the cleanest; bins lower on the vertical axis contain more noise) and length along the vertical axis is proportional to the number of times each bin was selected at a given step during training.

6.1 What did the agent learn?

Figure 3 shows a coarse visualization of the hand-optimized policy of Wang et al. (2018), adapted to our 6-bin scenario, compared to the Q-learning policy on the same scenario. The former, by design, telescopes towards the clean bins. Note that the latter policy is masked by the agent's exploration schedule, which slowly anneals toward nearly complete policy control, beginning at step 30,000. After this point, the learned policy takes over and continues to evolve. This learned policy has little in common with the hand-designed one. Instead of focusing on a mixture of the clean bins, it focuses on the cleanest bin and the second-to-noisiest. We hypothesize that returning to the noisy bin acts as a form of regularization, though this requires further study.

7 Conclusion

We have presented a method to learn a curriculum for presenting training samples to an NMT system. Using reinforcement learning, our approach learns the curriculum jointly with the NMT system during the course of a single NMT training run. Empirical analysis on the Paracrawl and WMT English-French corpora shows that this approach beats the uniform sampling and filtering baselines. In addition, we were able to match a state-of-the-art hand designed curriculum on Paracrawl and beat it on WMT.

We see this a first step toward enabling NMT systems to manage their own training data. In the future, we intend to improve our approach by eliminating the static exploration schedule and binning strategy, and extend it to handle additional data attributes such as domain, style, and grammatical complexity.

Acknowledgements

The authors would like to thank Wei Wang for his advice and help in replicating the CDS baselines.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017. Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 40–46.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86. Association for Computational Linguistics.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *The 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 678–683, Sofia, Bulgaria.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR*, abs/1612.06897.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1311–1320.

Tom Kocmi and Ondrej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. *arXiv preprint arXiv:1707.09533*.

Ming Liu, Wray L. Buntine, and Gholamreza Haffari. 2018. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, pages 334–344.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M Mitchell. 2019. Competence-based Curriculum Learning for Neural Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. 2017. Neural machine translation training in a multi-domain scenario. *arXiv preprint arXiv:1708.08712*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 130–139. Association for Computational Linguistics.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488. Association for Computational Linguistics.

Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 133–143. Association for Computational Linguistics.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410. Association for Computational Linguistics.

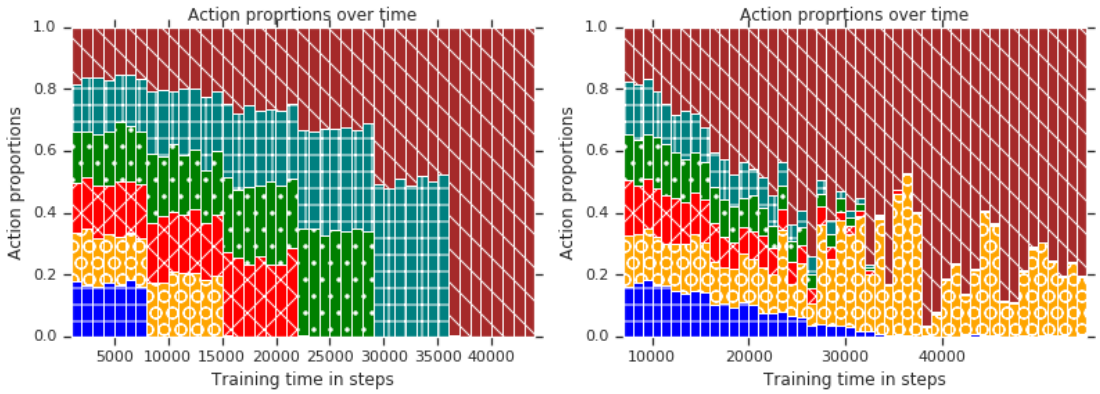
Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced co-training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1252–1262.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

A Appendix

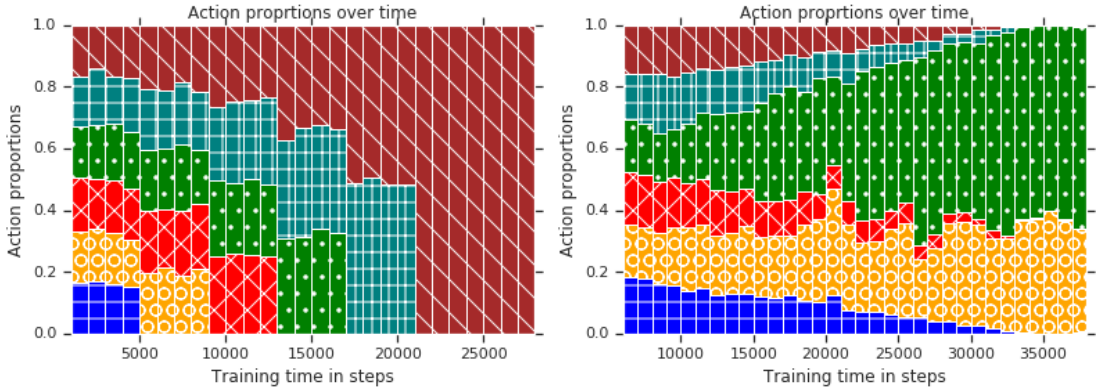
A.1 Q-learning hyper-parameters

- Observations: We sample 32 prototype sentences from each bin to create a *prototype batch* of 192 sentences.
- Q-networks: The two Q-networks were MLPs with 2 x 512-d hidden layers each. A tanh activation function was used.
- RL optimizer: We used RMSProp with a learning rate of 0.00025 and a decay of 0.95 and no momentum.
- NMT warmup : 5000 steps (no transitions from this period are recorded).
- Stack size: We do not stack our observations for the RL agent (i.e., stack size = 1).
- Exploration strategy : We use a linearly decaying epsilon function with decay period set to 25k steps. The decay floor was set to 0.01.
- Discount gamma : 0.99
- Update horizon : 2
- Minimum number of transitions in replay buffer before training starts: 3000



(a) Telescoping (b) RL learned

Figure 4: Policies learned by the RL agent on the Paracrawl En-Fr corpus compared against the telescoping policy from Wang et al. (2018). Lower bins on the vertical axis contain more noise.



(a) Telescoping (b) RL learned

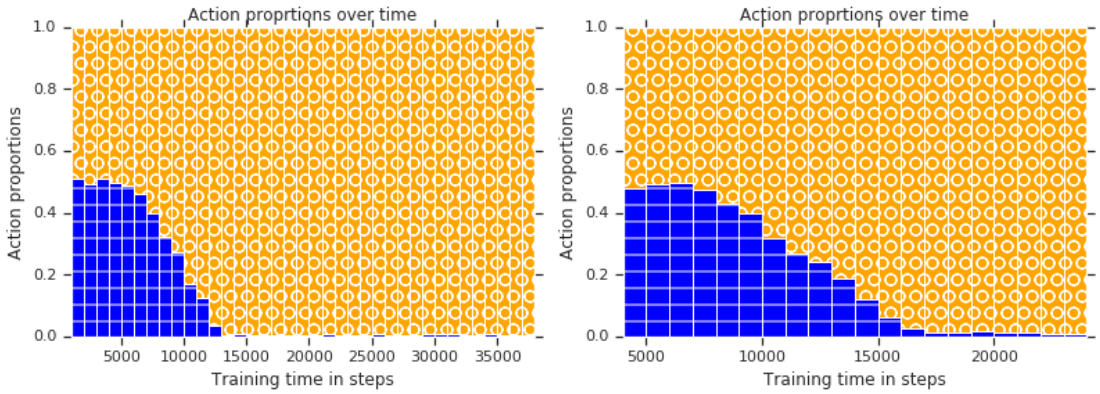
Figure 5: Policies learned by the RL agent on the WMT En-Fr corpus compared against the telescoping policy from Wang et al. (2018). Lower bins on the vertical axis contain more noise.

- Update period (how often the online Q-network is trained): 4 steps
- Target update period (how often the target Q-network is trained): 100 steps
- The window for the delta-perplexity reward was 1.

A.2 Learned Policies

Figures 4, 5 and 6 show coarse representations of the policies learned by the Q-learning agent on the Paracrawl and WMT English-French datasets. Each column in the figures represents the relative proportion of actions taken (bins selected) averaged over a thousand steps and the actions go from noisy to clean on the y-axis. Each policy starts from a uniform distribution over actions. Some salient aspects of the learned policies are listed below.

1. All learned curricula differ significantly from the hand-designed policies.
2. The RL curriculum learned for Paracrawl (Figure 4) focus on two bins during exploitation (choose action using the trained Q-function). Surprisingly, these are not the two cleanest bins but a mixture of the cleanest and the second-to-noisiest bin.
3. The RL curriculum learned for WMT (Figure 4) is closer to a uniform distribution over actions for a long duration. This makes sense since the data from WMT is mostly homogeneous with respect to noise. When the agent does decide to exploit some bins more often, they are not the cleanest ones, but the 1st and 4th bin instead.
4. Figure 6 shows the policies learned on the bookend task for Paracrawl and WMT; the



(a) RL Learned (Paracrawl)

(b) RL learned (WMT)

Figure 6: Policies learned by the RL agent on the 2-bin task on the Paracrawl and WMT En-Fr datasets. Lower bins on the vertical axis contain more noise.

only two bins available contain the noisiest and cleanest portion of the corpus. The RL agent very quickly learns that there is an optimal bin to choose in this task and converges to consistently exploiting it. We consider this a sanity check of curriculum learning methods.