META-LEARNING NEURAL MACHINE TRANSLATION CURRICULA

by

Gaurav Kumar

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

February, 2021

Abstract

Curriculum learning hypothesizes that presenting training samples in a meaningful order to machine learners during training helps improve model quality and convergence rate. In this dissertation, we explore this framework for learning in the context of Neural Machine Translation (NMT). NMT systems are typically trained on a large amount of heterogeneous data and have the potential to benefit greatly from curriculum learning in terms of both speed and quality. We concern ourselves with three primary questions in our investigation : (i) how do we design a task and/or dataset specific curriculum for NMT training? (ii) can we leverage human intuition about learning in this design or can we learn the curriculum itself? (iii) how do we featurize training samples (e.g., easy versus hard) so that they can be effectively slotted into a curriculum?

We begin by empirically exploring various hand-designed curricula and their effect on translation performance and speed of training NMT systems. We show that these curricula, most of which are based on human intuition, can improve NMT training speed but are highly sensitive to hyperparameter settings. Next, instead of using

ABSTRACT

a hand-designed curriculum, we *meta-learn* a curriculum for the task of learning from noisy translation samples using reinforcement learning. We demonstrate that this *learned* curriculum significantly outperforms a random-curriculum baseline and matches the strongest hand-designed curriculum. We then extend this approach to the task of *multi-lingual* NMT with an emphasis on accumulating knowledge and learning from multiple training runs. Again, we show that this technique can match the strongest baseline obtained via expensive fine-grained grid search for the (learned) hyperparameters. We conclude with an extension which requires no prior knowledge of sample relevance to the task and uses sample features instead, hence learning both the relevance of each training sample to the task and the appropriate curriculum jointly. We show that this technique outperforms the state-of-the-art results on a noisy filtering task.

ABSTRACT

Primary Readers and Advisors:

Dr. Sanjeev Khudanpur
Associate Professor,
Department of Electrical and Computer Engineering and
Department of Computer Science
Johns Hopkins University, Baltimore, MD

Dr. Philipp Koehn
Professor,
Department of Computer Science
Johns Hopkins University, Baltimore, MD

Secondary Reader:

Dr. Kevin Duh
Assistant Research Professor,
Department of Computer Science
Johns Hopkins University, Baltimore, MD

Acknowledgments

This thesis is the culmination of over four years of work with the direct support of many collaborators and advisors. Portions of this document appear in the following publications : X. Zhang, G. Kumar, et al. (2018), X. Zhang, Shapiro, et al. (2019), G. Kumar, Foster, et al. (2019), G. Kumar, Koehn, and Khudanpur (2021b) and G. Kumar, Koehn, and Khudanpur (2021a). This work would not have been possible without the effort of my co-authors: Xuan Zhang, Huda Kyrallah, Kenton Murray, Jeremy Gwinnup, Marainna Martindale, Paul McNamee, Kevin Duh, Marine Carpuat, Pamela Shapiro, George Foster, Colin Cherry, Maxin Krikun, Philipp Koehn and Sanjeev Khudanpur.

Sanjeev Khudanpur has been my advisor through my years as a Masters and Ph.D. student at JHU. Not only has he encouraged me to pursue my ideas through this time, while providing guidance and advice, I credit him with teaching me how to carry out research. I could not have asked for a better advisor. Thank you, Sanjeev.

Philipp Koehn has been my research advisor for the better part of four years. I am lucky to have had his calm influence and persistence in being there for me,

through both good and some really hard research times. He has been one of my closest collaborators through the majority of the work which appears in this thesis. Thank you, Philipp.

Chris Callison-Burch and Jason Eisner gave me my first exposure to research at the CLSP during my Masters program. Their guidance made my transition into Ph.D. research easier. Thank you, Chris and Jason.

Matt Post and Kevin Duh have been my research advisors and collaborators since the beginning of my time at JHU and I have been the benefactor of their very personal approach to mentoring and collaboration. Thank you, Matt and Kevin.

This work was conceived at MTMA 2018 under the guidance of Kevin Duh and Marine Carpuat. The next big portion of work was undertaken as a part of an internship at Google AI where I was mentored by George Foster and Colin Cherry. This formed the foundation for this thesis and I am grateful to George and Colin for hosting me and their guidance. I would like to thank Wei Wang for his advice and help in replicating baselines for this work.

I was fortunate to spend the early years of my Ph.D. life working on Speech Translation with Adam Lopez, Daniel Povey and Jan Trmal. My work with them led me to spend the later years working on Machine Translation. I would also like to thank the JSALT Machine Translation teams of 2015 and 2017 led by Chris Dyer and George Foster respectively. These were some of my best off-campus research experiences. I am also grateful to Graeme Blackwood, Abe Ittycheriah and Yaser

ACKNOWLEDGMENTS

Al-onaizan who hosted me at IBM research for an internship. Thank you.

I am thankful to my defense and GBO commitee for their valuable feedback and advice. Thank you, Sanjeev Khudanpur, Philipp Koehn, Kevin Duh, Jim Fill, James Spall and Paul Smolensky (alternate).

I want to thank my fellow CLSP students and alumni. Thank you, Adi Renduchintala, Tongfei Chen, Jaejin Cho, Shuoyang Ding, Katie Henry, Jonathan Jones, Huda Khayrallah, Ke Li, Ruizhi Li, Chu-Cheng Lin, Xutai Ma, Matthew Maciejewski, Matthew Weisner, Kelly Marchisio, Chandler May, Arya McCarthy, Hongyuan Mei, Sabrina Mielke, Phani Nidadavolu, Raghavendra Pappagari, Adam Poliak, Samik Sadhu, Elizabeth Salesky, Peter Schulam, Pamela Shapiro, Suzanna Sia, David Snyder, Brian Thompson, Tim Vieira, Yiming Wang, Zachary Wood-Doughty, Winston Wu, Patrick Xia, Hainan Xu, Jinyi Yang, Xuan Zhang, Ryan Cotterell, Naomi Saphra, Pushpendre Rastogi, Adrian Benton, Rachel Rudinger and Keisuke Sakaguchi, Yuan Cao, Matt Gormley, Ann Irvine, Keith Levin, Harish Mallidi, Vimal Manohar, Chunxi Liu, Courtney Napoles, Vijayaditya Peddinti, Nanyun Peng, Adam Teichert, and Xuchen Yao.

This work would not have been possible without the support of Ruth Scally, Yamese Diggs and Carl Pupa.

Finally, and most importantly, my family has been an incredible source of guidance and inspiration throughout my graduate life. Without their patience and support, none of this would have been possible. Thank you Mummy, Saurav, Shriya and Yashi.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The notion of a curriculum as it applies to human learning is a familiar one. Most formal instruction employs this to impose structure upon the task of learning, presenting concepts at different times and building upon existing knowledge to teach more complex abstractions. As an example, in the case of adults learning a new language, a human may be presented with the simplest nouns and verbs followed by short phrases before being asked to learn to form complete sentences or tackle other grammatical nuances (Richards, 1984). In the *animal training* literature, this concept is known as *shaping* (Skinner, 1958; Peterson, 2004; Krueger and Dayan, 2009).

The idea of whether such a curriculum would be useful to *machine* learners was first formally addressed by Bengio et al. (2009), who looked at the order of presenting examples during training to simple neural networks for the task of shape recognition and language modeling. They motivate and define this problem as:

> Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones. Here, we formalize such training strategies in the context of machine learning, and call them (the task of designing the order of presentation) "curriculum learning".

They defined the expected benefits from such a curriculum on machine learning as, (i) improved speed of convergence of the training process and (ii) improved performance and quality of the local minimum obtained post-training. Inspired by humans and the way they learn, most initial research (including Bengio et al. (2009)) in this field focused on presenting easy examples to the learning machine first and gradually increasing their complexity during training[1]. This however, presents the following questions:

Q1. What is an ideal curriculum for training a machine learning model and how does this change based on the properties of the task and the dataset? Can the curriculum itself be learned, instead of relying on hand-designed versions?

Q2. Do we expect human intuition about learning via curricula to translate to machine learning? Specifically, does training of models stand to gain (performance or speed) from knowledge of how humans learn through gradual exposure from easy to more complex examples?

Q3. How does one define the notion of an easy or hard example with respect to the training of machine learning models? Do human-designed heuristics for

[1] A related approach was to start with a simple model and gradually increase the model complexity/capacity as training went on.

Task	Feature:Value	Translation Pair (Source, Target)
Denoising	Noise: Clean	Frau Präsidentin, zur Geschäftsordnung. Madam President, on a point of order.
	Noise: Noisy	das Bild zeigt die originale Pfeife! 28 . November 2015
Domain Adaptation (Patent)	Domain: Subtitles	是他完全迷上我了 Yeah, he totally has a crush on me.
	Domain: Patent	显示装置及显示系统 In-cell touch device and implementation method thereof
Multi-lingual training	Language ID: De->En	Sie engagieren sich wirklich für die Themen, die Sie bewegen. You actually put action to the issues you care about.
	Language ID: Fr->En	Vous agissez en fait sur les questions qui vous intéressent You actually put action to the issues you care about.

Table 1.1: Examples of heterogeneous translation pairs from various machine translation tasks.

determining sample *difficulty* work and is it possible to do away with the need for such heuristics?

The work presented here attempts to seek answers to these questions in the context of Neural Machine Translation (NMT) (Sutskever, Vinyals, and Quoc V Le, 2014). Using artificial neural networks, NMT attempts to learn models which can translate sentences (sequences of words) from one language (source) to another (target). These models are hence referred to as *translation models*. NMT is a good test case for curriculum learning, as training is very computationally expensive in large data conditions required to reach good performance. As stated earlier, presenting the right samples to the machine learner (NMT in this case) at the right time during (NMT) training may help improve convergence speed of training. Additionally, training data for large-scale NMT models are typically very heterogeneous (varying in characteristics such as domain, translation quality, and degree of linguistic difficulty), are often

derived from noisy web-crawls, and can contain hundreds of millions of sentence pairs (see Table 1.1 for examples), not all of which may be useful or non-redundant. This reinforces the idea that all training samples are not equal when translation performance or training time is the metric to optimize. Finally, as with other machine learning applications which encounter similar problems, some hand-designed curricula have shown significant improvement in translation performance of NMT systems. The most commonly encountered and extreme versions are *data filtering* (Moore and William Lewis, 2010; Axelrod, He, and Gao, 2011b; Duh et al., 2013; Durrani et al., 2016), only exposing the model to a selected portion of the dataset during training, and *fine-tuning* (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016), training the model to convergence on one subset of the dataset and then further training this (converged) model on another carefully chosen subset.

The questions (Q1-Q3) posed above with respect to curriculum learning in general apply to NMT training quite naturally. First, given the scale and the heterogeneous nature of the training data, it is hard to apply human intuition to design an optimal curriculum. Most state-of-the-art NMT systems rely on the volume of the training data or filtering heuristics to explain away the differences in the training samples. Second, it is not clear how to quantify sample difficulty or *usefulness* to improve translation performance. This problem is compounded by the fact that an NMT model's parameters change during training and a sample which may be considered difficult to learn from at the beginning of training may not remain the same after a few

model updates. The most commonly used proxies for sample difficulty rely on knowledge from an auxiliary *trusted* model, linguistic features of the training sample, and sentence-pair quality metrics. It is also unclear whether a single usefulness heuristic is sufficient or if more than one should be used in a complementary manner. Third, designing a optimal curriculum (the order in which examples are presented to the NMT system), would involve a combinatorial search over all orderings of the training data. This is obviously computationally intractable and a tractable way to search over the space of orderings is required. Finally, NMT systems, for computational reasons, already rely on established data organization methods e.g., sorting samples to make sure samples in a mini-batch have roughly the same length, to deal with the scale and varying length of training samples, and it is not clear how a curriculum should interact with these existing design decisions.

In this dissertation, we conduct a systematic study of these issues related to curriculum learning in the context of NMT training. We start in Chapter 3 by conducting an empirical examination of various hand-designed curricula and their effect on translation performance and training speed of NMT systems. We explore difficulty criteria based on auxiliary NMT model scores as well as linguistic properties. We consider a wide range of schedules, based not only on the easy-to-difficult ordering, but also on strategies developed independently from curriculum learning, such as dynamic sampling and boosting (D. Zhang et al., 2017; Wees, Bisazza, and Monz, 2017a; R. Wang, Utiyama, and Sumita, 2018). Using a probabilistic view of curriculum learning

which can accommodate both fixed and dynamic curricula, we propose modifications to the data sampling component of NMT training which are modular with respect to the optimizer of the NMT system Our experiments on a German-English translation task confirm that curriculum learning can improve convergence speed without loss of translation quality, and show that viewing curriculum learning more flexibly than strictly training on easy samples first has some benefits. We also demonstrate that hand-designed curricula are highly sensitive to hyperparameters, and no single strategy emerges as clearly or uniformly the experiments.

Since designing and setting hyper-parameters to specify a curriculum is usually a matter of extensive trial and error, automating this process with *meta-learning*[2] is an attractive proposition. Our next line of inquiry in Chapter 4, focuses on *meta-learning* a curriculum for the task of training an NMT system on extremely noisy French-English and German-English datasets. We attempt to match the performance of a state-of-the-art non-trivial reference curriculum proposed by W. Wang, Watanabe, et al. (2018a), in which training gradually focuses on increasingly cleaner data, as measured by an external scoring function. To effectively search through the large space of possible curricula, we use a reinforcement-learning (RL) approach involving a learned agent whose task is to select data representing a given noise level, at each NMT training step to optimize eventual translation performance. We demonstrate that this

[2]We note that *meta-learning* is a loaded term in machine learning research. Recently, it has been associated with model selection and hyperparameter tuning (e.g. autoML) of machine learning models. Multi-task learning has also been framed as a *meta-learning* problem. In our work we focus on methods which decide on which training samples are most appropriate to train on by a machine learner given the task and the properties of the dataset.

approach can learn a curriculum which significantly outperforms a random-curriculum baseline and match the performance of the strongest hand-designed curriculum. Interestingly, it does so using a different strategy from the best hand-designed curriculum.

Most hand-designed curricula do not change as the parameters of an NMT system evolve during training. As stated earlier, this can be a disadvantage since the optimal samples to expose the model to may change as it evolves. We address this problem in chapter 5 to addressing this problem by learning time-varying curricula from multiple NMT training runs. We do this in the context of *multi-lingual* NMT training (an instance of *multi-task* training) where datasets from two language pairs, Nepali-English and Hindi-English in our case, are used in conjunction to improve the translation performance for one or both of the language pairs. We use pruned-tree search and multi-arm bandits to learn these dynamic curricula. The task of the bandit is to condition on the state of the NMT model and learn if it is appropriate to expose the model to samples from one language pair or the other. Our experiments show that these learned curricula can match the performance of the strongest fixed curricula (obtained through a coarse and expensive grid search) but are themselves computationally expensive to obtain, because effectively conditioning on temporal observations requires the multi-arm bandit to collect samples from many NMT training runs.

As discussed earlier, determining sample usefulness for training can be a hard task. Chapter 6 describes our effort to do away with an explicit notion of usefulness

	Sample Usefulness metric	Time Dependent?	Curriculum Type	Secondary Task
Empirical Exploration (Chap. 3)	Auxiliary model scores, linguistic features	No	Hand-Designed	Domain Adaptation
RL-Q-Learning (Chap. 4)	Noise CDS scores	Yes	Learned	Denoising
Multi-lingual bandits (Chap. 5)	Language ID	Yes	Learned from multiple runs	Multi-lingual translation
Reward modeling (Chap. 6)	Learned based on features	No	Learned from multiple runs	Filtering

Table 1.2: A summary of the work on learning curricula for NMT training.

and instead backs off to representing each sample as a set of features that may be correlated to usefulness. Using feedback from the training of multiple NMT systems, we instead *learn* an interpolation of the features which serves as a score to define a fixed filtering-based curriculum. Hence, even though the *type* of the curriculum is fixed in this case, its configuration which relies on the interpolated features is learned and requires no prior knowledge about which features are informative or even if they are mutually redundant. Our experiments, which apply this method to building NMT systems for a noisy Estonian-English dataset, show that it outperforms a strong single-feature filtering-curriculum and hand-designed feature interpolation. Additionally, we show that this method is robust in the presence of the kinds of noise most prevalent in web-crawled datasets.

Table 1.2 provides a summary of the work which appears in this dissertation. In the next chapter, we start this investigation with a survey of prior work most closely related to this dissertation.

Chapter 2

Related Work

Curriculum Learning research has its roots in cognitive science, especially as it relates to language learning in humans. One of the earliest lines of inquiry looked at how presenting positive and negative samples affected language learners and whether the order in which they are presented can affect learning itself (Gold, 1967; Baker, 1979; Bowerman, 1987; Pinker, 1989; Wexler and Cullicover, 1980). Building from this Newport (1988), Newport (1990), and Plunkett and Marchman (1990) looked at the effect of gradually increasing the size of the input data to learners during learning. Meanwhile, Ash (1989), Fahlman and Lebiere (1990), and Shultz and Schmidt (1991) were examining the effect of gradually increasing the size of learning models during training. It is also worth noting that curriculum learning is a fairly prominent field of inquiry in education research where it is also known as *shaping* (Skinner, 1958; Peterson, 2004; Krueger and Dayan, 2009).

In the field of machine learning, Elman (1993), Rohde and Plaut (1994), and Krueger and Dayan (2009) first looked at the effect of *starting small*, that is, starting learning with easier concepts and then gradually increasing the difficulty of the concepts. Krueger and Dayan (2009) looked at the effect of gradually increasing the capacity of machine learners during training while Sanger (1994) explored similar frameworks in the field of robotics. Bengio et al. (2009) coined the term of *curriculum learning* to generally refer to techniques which guide the training of learning systems "by choosing which examples to present and in which order to present them in the learning system", and hypothesized that training on easier samples first is beneficial. In their work, they improve neural language model training using a curriculum based on increasing vocabulary size.

In the field on NLP, organizing training samples based on difficulty has shown improvements in performance outside of neural models e.g., Spitkovsky, Alshawi, and Jurafsky (2010) bootstrap unsupervised dependency parsers by learning from incrementally longer sentences. Curriculum learning has also gained popularity to address the difficult optimization problem of training deep neural models (Bengio, 2012). More recently, Tsvetkov et al. (2016) improve word embedding training using Bayesian optimization to order paragraphs in the training corpus based on a range of distributional and linguistic features (diversity, simplicity, prototypicality).

While curriculum learning often refers to organizing examples from simple to difficult, other data ordering strategies have also been shown to be beneficial: Amiri,

Miller, and Savova (2017) improve the convergence speed of neural models using spaced repetition, a technique inspired by psychology findings that human learners can learn efficiently and effectively by increasing intervals of time between reviews of previously seen materials. Curriculum design is also a concern when deciding how to schedule learning from samples of different tasks either in a sequence from simpler to more difficult tasks (Collobert and Weston, 2008) or in a multi-task learning framework (Graves, Bellemare, Menick, Rémi Munos, et al., 2017a; Kiperwasser and Ballesteros, 2018b).

In the sub-field of Neural Machine Translation (NMT), the exploration of curriculum learning is in its infancy. In practice, training protocols randomize the order of sentence pairs in the training corpus (Sennrich, Firat, et al., 2017; Hieber et al., 2017). There are works that speed training up by batching the samples of similar lengths (Khomenko et al., 2016; Doetsch, Golik, and Ney, 2017). Such works attempt to improve the *computational efficiency*, while curriculum learning is supposed to improve the *statistical efficiency* — fewer batches of training examples are needed to achieve a given performance.

Kocmi and Bojar (2017) conducted the first study of curriculum learning for NMT by exploring the impact of several criteria for curriculum design on the training of a Czech-English NMT system for one epoch. They ensure samples within each mini-batch have similar linguistic properties, and order mini-batches based on complexity. They show translation quality can be improved by presenting samples from easy to

hard based on sentence length and unigram frequency.

Previous work has also investigated dynamic sampling strategies, emphasizing training on samples that are expected to be most useful based on model scores or domain relevance. Inspired by boosting (Schapire, 2002), D. Zhang et al. (2017) assign higher weights at each epoch to training examples that have lower perplexities under the model of the previous epoch. Similarly, Wees, Bisazza, and Monz (2017a), and R. Wang, Utiyama, and Sumita (2018) improve the training efficiency of NMT by dynamically selecting different subsets of training data between different epochs. The former perform this dynamic data selection according to domain relevance (Axelrod, He, and Gao, 2011a) while the latter use the difference between the training costs of two iterations.

There have been several efforts to specifically derive hand-designed curricula for the task of dealing with heterogeneous data, a typical characteristic of the data available for training NMT systems. The current approaches fall under two broad (but related) categories depending on whether we choose to use all of the available training data or only a part of it. *Data Selection* considers the problem of finding the a subset of the training data to train on with the aid of an auxiliary model which is closer to dataset with known characteristics of interest (Moore and William Lewis, 2010; Axelrod, He, and Gao, 2011b; Duh et al., 2013; Durrani et al., 2016). *Curriculum* based approaches hand-design an ordering over samples to determine how they are presented to the NMT model during training. The current state-of-the-art NMT

systems typically fall in the latter category.

For training on noisy data, W. Wang, Watanabe, et al. (2018a) propose an approach which begins training on all of the available data and gradually eliminates noisy samples so that we end up training on a clean subset of the training data. Wees, Bisazza, and Monz (2017b) had previously proposed a similar approach for the task of domain adaptation for NMT. Jean, Firat, and M. Johnson (2018) propose explicit curricula for multilingual NMT training. Continued training (also known as fine-tuning) (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016) is another popular curriculum-based approach when we have a small amount of training data with the attribute of interest. For example, in the case of domain adaptation, the approach is to train on the large out-of-domain subset of the training data to convergence and then train on the small in-domain subset. Previous work on dealing with heterogeneous data in NMT includes approaches that modify parts of the model (Dakwale and Monz, 2017; Kobus, Crego, and Senellart, 2016; Britz, Q. Le, and Pryzant, 2017). Broadly construed, it extends also to multi-task (Kiperwasser and Ballesteros, 2018a) and multilingual (M. Johnson, Schuster, Quoc V Le, et al., 2017a; Gu et al., 2018) scenarios.

Instance weighting (Matsoukas, Rosti, and B. Zhang, 2009; Shah, Barrault, and Schwenk, 2010; Foster, Goutte, and Kuhn, 2010; R. Wang, Utiyama, L. Liu, et al., 2017) is another instance of curriculum-based approaches where each sentence is weighted based on predefined functions, and this weight is incorporated into the

training procedure or is used to determine how the instances are sampled.

Techniques that re-weight (B. Chen et al., 2017) or re-order examples to deal with domain mismatch (Wees, Bisazza, and Monz, 2017b; Sajjad et al., 2017) or noise (W. Wang, Watanabe, et al., 2018a) are also related to this work. Additionally, recent work has explored bandit optimization for scheduling tasks in a multi-task problem (Graves, Bellemare, Menick, Rémi Munos, et al., 2017b), and reinforcement learning for selecting examples in a co-trained classifier (J. Wu, Li, and W. Y. Wang, 2018). Techniques related to active learning are also relevant to this work. A typical setting for active learning for NMT is as follows: given a pool of unlabeled (monolingual) source text, we need to find the most useful sentences to query their translation from an oracle. In our setting, this setting could be modified to instead select the most useful sentences which optimize translation quality. M. Liu, Buntine, and Haffari (2018) and Peris and Casacuberta (2018) apply imitation learning in an active learning framework to actively select monolingual training sentences for labeling in NMT, and show that the learned strategy from one language-pair can be transferred to a related language-pair.

Filtering and denoising are two simple and popular curriculum-based approaches which have shown consistent utility in NMT training (speed and quality). These mostly focus on pre-filtering using hand-crafted rules and on using sentence pair scoring and filtering methods. Deterministic hand-crafted rules (Hangya and Fraser, 2018; Kurfalı and Östling, 2019) remove sentence pairs with extreme lengths, unusual

sentence length ratios and exact source-target copies, and are extremely effective in removing most of the obvious extraction errors in automatically extracted datasets. Automatic sentence pair scoring functions have been used successfully to filter noisy corpora as well. This includes the use of language models (Rossenbach et al., 2018), neural language models trained on trusted data (Junczys-Dowmunt, 2018) and lexical translation scores (González-Rubio, 2019). Chaudhary et al. (2019) propose the use of cross-lingual sentence embeddings for determining sentence pair quality while several efforts (Kurfalı and Östling, 2019; Soares and Costa-jussà, 2019; Bernier-Colborne and Lo, 2019) have focused on the use of monolingual word embeddings. Parcheta, Sanchis-Trilles, and Casacuberta (2019) use a machine translation system trained on clean data to translate the source sentences of the noisy corpus and evaluate the translation against the original target sentences using BLEU scores. Erdmann and Gwinnup (2019) and Sen, Ekbal, and Bhattacharyya (2019) propose similar methods using METEOR scores and Levenshtein distance respectively. Rarrick, Quirk, and Will Lewis (2011), Venugopal et al. (2011) and Antonova and Misyurev (2011) present techniques for detecting machine translated sentence pairs in corpora. Tools such as LASER (Schwenk and Douze, 2017), BiCleaner (Sánchez-Cartagena et al., 2018) and Zipporah (Xu and Koehn, 2017) have been used (Chaudhary et al., 2019) for noisy corpus filtering. Curriculum learning has been used to obtain policies for data selection that can expose the model to noisy samples less often during training (W. Wang, Watanabe, et al., 2018b; G. Kumar, Foster, et al., 2019). More recently,

ElNokrashy et al. (2020) and Esplà -Gomis et al. (2020) have used classifier based approaches to filtering noisy parallel data.

In recent work related to curriculum learning, Zhou et al. (2020) use the training model's cross-entropy for a sample as a proxy for its difficulty for the task while X. Liu et al. (2020) use the norm of the word embeddings of a sentence as a difficulty measure. W. Wang, Caswell, and Chelba (2019) use an expectation maximization style procedure for learning dynamic data selection policies for training with noisy data. X. Wang et al. (2019) use a reinforcement learning framework to learn a reward model as a function of the training data and show gains on machine translation tasks. Kreutzer and Riezler (2019) apply curriculum learning to an interactive sequence to sequence learning problem while Wan et al. (2020) use self paced learning (M. Kumar, Packer, and Koller, 2010) for designing NMT curricula.

Taken together, these prior works show that curriculum learning has the potential to improve the training speed and quality of machine translation models. However, especially for NMT, these curricula are hand-designed and extremely sensitive to hyperparameters. Thus, *meta-learning* these curricula jointly with the task of training the NMT system, can be an attractive proposition.

Chapter 3

An Empirical Exploration of Curriculum Learning for Neural Machine Translation

Curriculum learning hypothesizes that choosing the order in which training samples are presented to a machine learner can increase performance of the learned model on the chosen task and improve training convergence speed. In particular, presenting samples that are easier to learn from before presenting difficult samples is an intuitively attractive idea, which has been applied in various ways in Machine Learning and Natural Language Processing tasks (Bengio et al., 2009; Tsvetkov et al., 2016; Cirik, Hovy, and Morency, 2016; Graves, Bellemare, Menick, Rémi Munos, et al., 2017a, inter alia). In addition, other human-learning derived concepts in designing

curricula such as reinforcing via repetition may aid machine learning.

We start our study of curriculum learning by performing an extensive empirical examination of various hand-designed curricula and their effect on translation performance and training speed of NMT systems. While NMT systems may benefit from improvement in either one of these aspects, designing a curriculum for NMT training can be a complex problem. As discussed in Chapter 1, even if we hand-designed curricula based on the assumption that machines can learn in a manner similar to humans, we are left with the problem of quantifying sample difficulty or more generally, *usefulness*.

In this chapter, we seek answers to the following questions:

1. Do hand-designed curricula derived from human intuition help train better NMT systems faster? If so, is there a consistent best strategy?

2. Are there proxies for sample difficulty or *usefulness* which help?

We adopt a probabilistic view of curriculum learning that lets us explore a wide range of curricula flexibly. Our approach does not impose deterministic selection from samples. Instead, each sample has a probability of being selected for training, and this probability changes depending on the difficulty of the sample and on the curriculum's schedule. This framework allows us to work with both deterministic and stochastic curricula. We explore difficulty criteria based on auxiliary NMT model scores as well as linguistic properties. We consider a wide range of schedules, based not only on

the easy-to-difficult ordering, but also on strategies developed independently from curriculum learning, such as dynamic sampling and boosting (D. Zhang et al., 2017; Wees, Bisazza, and Monz, 2017a; R. Wang, Utiyama, and Sumita, 2018).

We begin by describing our probabilistic approach to curriculum learning. Next, we examine auxiliary model-based and linguistic criteria for determining sample usefulness. Using these, we proceed to look at various hand-designed curricula, building upon previous work in curriculum learning. We apply these to the task of learning on a German-English translation task and examine if these curricula can improve convergence speed without loss in translation quality. We conclude by analyzing the trends from applying hand-designed curriculum learning to NMT and by summarizing our findings. The work described below appears in X. Zhang, G. Kumar, et al. (2018)[1] and X. Zhang, Shapiro, et al. (2019).

3.1 A Probabilistic View of Curriculum Learning

Let (x, y) be a bitext example, where x is the source sentence and y is the target reference translation. We use subscripts i to denote the sample index and assume a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1,2,...S}$ of size S. Curriculum learning can then be formu-

[1]The author of this dissertation participated in the conception of this approach, designing and running experiments and writing this paper along with the co-authors on this paper. This work was performed by a team of researchers (co-authors on the paper) at the Machine Translation Marathon 2018 in Pittsburgh.

lated in a probabilistic manner, where each sentence pair (x_i, y_i) has a probability of being selected for training, and this sampling probability changes depending on the difficulty of the example and the curriculum schedule (Bengio et al., 2009).

Specifically, we divide the curriculum schedule into distinct phases t which correspond to different time points during training. For instance, $t = 1$ could be the first N checkpoints, $t = 2$ is the next N checkpoints, etc. The definition of phases is flexible: alternatively $t = 1$ may correspond to the first epoch, and $t = 2$ may correspond to the second epoch (or more). At each phase t, we maintain a multinomial distribution q_i^t over the examples in $\mathcal{D}$, where $q_i^t \geq 0 \ \forall i$ and $\sum_{i=1}^{S} q_i^t = 1$. To implement the curriculum schedule that begins with easy examples, we would start at $t = 1$ by setting q_i^t to be high for easy examples and q_i^t to be low (or zero) for difficult examples. Gradually, for large t, we increase q_i^t for the more difficult examples. At some point, all examples have equal probability of being selected; this corresponds to the standard NMT training procedure. An illustration of this probabilistic view of curriculum learning is shown in Figure 3.1.

There are two advantages to this probabilistic sampling view of curriculum learning:

1. It is a flexible framework that enables the design of various kinds of curriculum schedules. By specifying different kinds of distributions, one can perform easy-to-difficult training or the reverse difficult-to-easy training. One can default to uniform sampling, which corresponds to standard training with random mini-

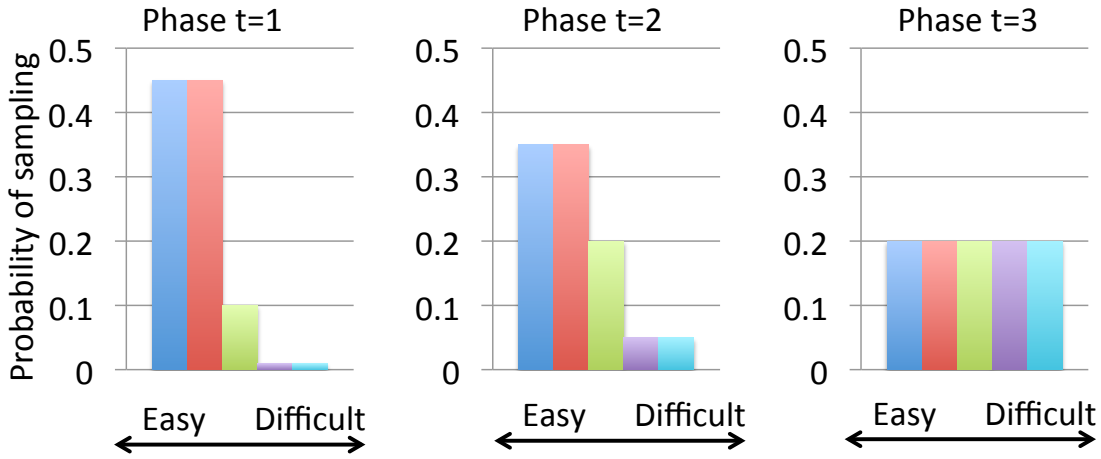


Figure 3.1: Probabilistic view of curriculum learning: On the x-axis, the examples are arranged from easy to difficult. y-axis is the probability of sampling the example for training. By specifying different kinds of sampling distributions at different phases, we can design different curriculums. In this example, $t = 1$ samples from the first three examples, $t = 2$ includes the remaining two examples but at lower probability, and $t = 3$ defaults to uniform sampling (regardless of difficulty).

batches. Many of these variants are described in Section 3.3.2. Additionally, this framework will serve us in subsequent chapters where we explore curricula that are learned from scratch.

2. It is simple to implement in existing deep learning frameworks, requiring only a modification of the data sampling procedure. In particular, it is modular with respect to the optimizer's learning rate schedule and mini-batch shuffling mechanism; these represent best practices in deep learning, and may be suboptimal if modified. Further, the optimizer only needs access to sampling probability q_i^t, which abstracts away from the various selection criteria such as sentence length and vocabulary frequency (to be described in Section 3.2). This enables us to

plug-in and experiment with many criteria.

Without loss of generality, in practice we group examples into *shards* (Figure 3.2) such that those in the same shard have similar difficulty criteria.[2] Then we define the sampling distributions over shards rather than examples. Since there are fewer shards than examples (e.g., 5 shards vs. 1 million examples for a typical-sized dataset), the distributions are simple to design and visualize. Sharding is described in more detail in Section 3.3.1.

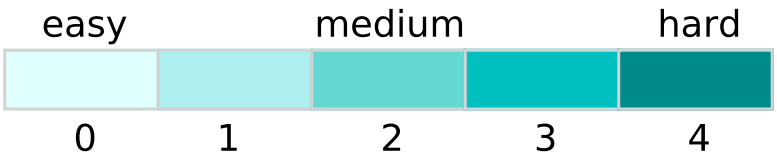


Figure 3.2: Training data organized by level of difficulty. Each block is a shard (i.e., a subset of the dataset) and darker shades indicate increasing difficulty. Note that the width of each patch does *not* indicate the number of samples in that shard, as it may vary for different difficulty criteria.

3.2 Sample Difficulty Criteria

In this work, we quantify the translation difficulty of a sentence pair (training sample for NMT systems) by two kinds of criteria (or scores): 1) how well an auxiliary translation model rates the pair and 2) linguistic features which are orthogonal to any translation model.

[2]Shards are not to be confused with buckets (grouping of similar-length samples). Shards are simply subsets of the training data and may allow for bucketing by length within themselves.

3.2.1 Model-based Difficulty Criteria

Given a training sample (x, y) its *one-best score* is the probability of the one-best translation (the product of its word prediction probabilities) from an auxiliary (possibly simpler) translation model, given a source sentence. This represents $p(\hat{y} \mid x)$ which is the probability of the one-best translation $\hat{y}$ given the source sentence x, as specified by the auxiliary model. A high *one-best score* for a translation suggests that the auxiliary model is very certain of its prediction with a small chance of choosing other candidates. Although the prediction might not be the "correct answer", $p(\hat{y} \mid x)$ represents the confidence of the model for that prediction, and indicates how easy the prediction is according to the auxiliary translation model.

3.2.2 Linguistic Difficulty Criteria

Linguistic features, including sentence length and vocabulary frequency, can also be used to measure the difficulty of translating a sample (Kocmi and Bojar, 2017). Short sentences usually do not have difficult syntactic structures, while lengthier sentences with long-distance dependencies are difficult to handle for NMT models (Hasler et al., 2017). To capture this phenomenon, we rank samples by the lengths of the source and target sentences and by the sum of the length of each sentence in the pair.

Sutskever, Vinyals, and Quoc V Le (2014) show that a NMT model's performance

decreases on sentences with more rare words. Similar to Kocmi and Bojar (2017), we first sort words by their frequency to get the word frequency rank, then order sentences based on the rank of the least frequent word in the sentence (*max word frequency rank*). Organizing sentences by this criterion is equivalent to gradually increasing the vocabulary size and training on sentences that only contain words in the current partial vocabulary (Bengio et al., 2009). In addition to this, we also experimented with the *average word frequency rank*. Again, we collect word frequency rank scores for source sentences, target sentences and concatenations of both[3].

3.3 Methods

Having defined criteria for measuring sample difficulty and illustrated how they can be used in a probabilistic curriculum learning framework, we now describe in more detail how this framework was instantiated for our study. We present our approach for organizing data into shards given sample difficulty scores (Section 3.3.1), how the shards are used by the curriculum schedule (Section 3.3.2), and how this fits in the overall training strategy (Section 3.3.3).

[3]In the concatenation, the word rank is obtained based on whether the word belongs in the source or the target; i.e., we maintain separate word frequency lists for each language.

3.3.1 Data Sharding

As described in section 3.1, samples are grouped into shards of similar difficulty (Figure 3.2). This can be done by various methods. One approach is to set thresholds on the difficulty score (Kocmi and Bojar, 2017). An alternative is to distribute the data evenly such that each shard will have same number of samples. The first approach makes it difficult to choose reasonable breaks while trying to ensure that each shard has roughly the same number of samples (Figure 3.3). In contrast, the latter may result in unwanted fluctuations in difficulty within the same shard, and not enough difference between different (especially adjacent) shards.

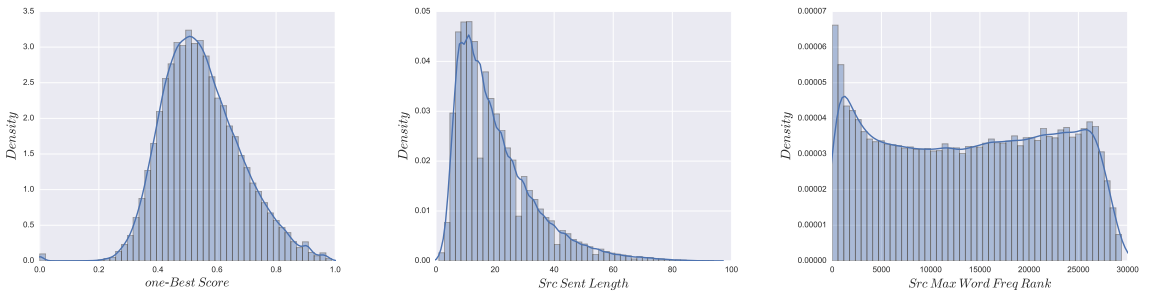


Figure 3.3: Difficulty score distribution on DE-EN TED Talks training set (151,627 sentence pairs in total) scored by three different difficulty criteria mentioned under the x-axis. Sharding results generated from Jenks Natural Breaks classification algorithm are shown below each subplot, in the ascending order of difficulty levels.

We instead use the Jenks Natural Breaks classification algorithm (Jenks, 1997), an algorithm commonly used in Geographic Information Systems (GIS) applications (Brewer, 2006; Chrysochoou et al., 2012). This method seeks to minimize the variance within classes and maximize the variance between classes. Figure 3.3 shows

examples of the univariate classification results using Jenks algorithm on our training corpus (TED Talks, Duh (2018)) where training samples are reorganized by three criteria (one-best score, source sentence length and source max word frequency rank) representing difficulty (see Section 3.2). Distributions obtained for other complexity criteria are available in the supplementary material.

3.3.2 Curriculum Schedule

The curriculum's *schedule* defines the order in which samples of different difficulty classes are presented to the learning system. A curriculum's *phase* is the period between two curriculum updates.[4] For NMT models, it is natural to consider the idea of first presenting easy samples to the models. In the following sections, we refer to this as the *default* schedule. We also introduce four variants of the *default* schedule (Figure 3.4) which lets us explore different trade-offs.

- **default** Shards are sorted by increasing level of difficulty. Training begins with the easiest shard and harder shards are included in subsequent phases.
- **reverse** Shards are sorted in descending order of difficulty. Training begins with the hardest shard and easier shards are included in subsequence phases.
- **boost** A copy of the hardest shard is added to the training set, after the model has processed shards of all difficulty classes.

[4]This is similar to the concept of an epoch except that only a subset of the training data may be available based on the curriculum's schedule.

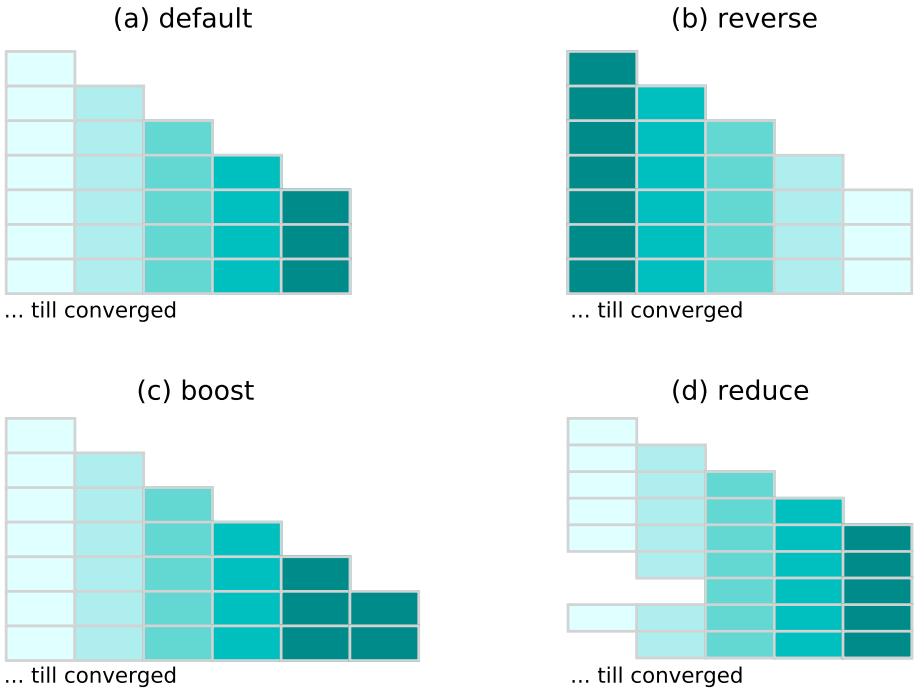


Figure 3.4: Training with different curriculum schedules. The colored blocks are shards of different difficulty levels (see figure 3.2). Within a sub-figure, each row represents a phase, and shards in that row are accessible shards based on the curriculum. Training starts from the first row and goes through the following rows in succession. Hence, at each phase only subsets of the training data and certain difficulty classes are available. Note that shards (and the samples within them) are shuffled as described in Section 3.3.3.

- **reduce** Once all shards have been visited, we start removing shards from training, one shard at the end of each phase, starting with the easiest. Once a fixed number of shards have been removed (2 in our case), we add them back. This *reduce and add-back* procedure will be iteratively continued until the training converges. The effect is that the model gets to look at harder shards more often.

- **noshuffle** Same as *default* except that shards are never shuffled; that is, they are always presented to the model in ascending order of difficulty (Samples within shards are shuffled as usual).

The *reverse* schedule tests the assumption that presenting easy examples first helps learning. It remains unclear if we should start with the easier sentences and move to more difficult ones, or if perhaps some of the difficult sentences are too hard for the model to learn and we should focus on straightforward sentences at the end. In addition, we are unsure of what the model will find more easy or difficult.

Another open question is whether presenting shards randomly during each curriculum phase (as done in the *default* schedule) weakens the curriculum. We explore an alternative by forcing the shard visiting order to be deterministic — always starting from the easiest shard, ending at the hardest shard for this phase. We label this schedule as *noshuffle*, since shuffling does not occur. *Noshuffle* may be helpful in the sense that every time the model is assigned with a new harder shard, it will review old shards in a more organized way. This method can be viewed as restarting the curriculum at each phase.

The last two schedules are adapted from D. Zhang et al. (2017), who improve NMT convergence speed by duplicating samples considered difficult based on model scores. The *boost* schedule combines the idea of training on easy samples first (from *default*), while putting more emphasis on difficult samples (as in *reverse*). The *reduce* schedule additionally makes sure that the model gets to look at difficult shards more

often. This is accomplished by periodically removing easy shards from epochs, and adding them back again later, also periodically.

3.3.3 Training Strategy

Finally, we address the question of how to draw mini-batches from the training data which has been sharded based on difficulty. Current state-of-the-art NMT model implementations *bucket* the training samples based on source and target length. Mini-batches are then drawn from these buckets, which are shuffled at each epoch. One way of drawing mini-batches while conditioning on difficulty is to sort the training samples by difficulty and to then draw these deterministically starting from the easiest to the most difficult sample. However, this loses the benefits gained by shuffling the data at each epoch.

Instead, our work uses a strategy similar to the work of Bengio et al. (2009). We organize samples into shards[5] according to the univariate classification results (Section 3.3.1) and allow further bucketing by sentence length within each shard. Samples within each shard are shuffled at each epoch, ensuring that we draw random mini-batches of the same difficulty.

Given shards of different difficulty levels, we follow these steps for training:

- The curriculum's schedule defines which shards are available for training. We call these the *visible* shards for this phase of NMT training.

[5] 5 shards in our experiments.

- These shards are then shuffled (except when we use the *noshuffle* schedule)[6] so that the model is trained using random levels of difficulty (in contrast to always using easy to hard).

- The samples within each shard are shuffled and bucketed by length. Mini-batches are drawn from these buckets.

- When the *curriculum update frequency* is reached (defined in terms of number of batches), the curriculum's schedule is updated. For example, this may imply that we include more difficult shards in training in the next phase or eliminate an existing shard. In cases where the total number of examples in these shards is smaller than the curriculum update frequency, we repeat the previous step until the update frequency has been achieved.

- After all available shards have been exposed to the model, training continues until validation perplexity does not improve for 32 checkpoints. The NMT model has then *converged*.

[6]In shuffling, we ensure that the first shard for this phase is not the same as the last shard from the last phase.

3.4 Experiment Setup

3.4.1 Data

All experiments in this study were conducted on the German(de)-English(en) parallel dataset (de-en) from the Multi-target TED Talks Task (MTTT) corpus (Duh, 2018). The *train* portion consists of about 150k parallel sentences while the *dev* and *test* subsets have about 2k sentences each. All subsets were tokenized and split into subwords using byte pair encoding (BPE) (Sennrich, Haddow, and Birch, 2016). The BPE models were trained on the source and target language separately and the number of BPE symbols was set to 30k per language.

3.4.2 NMT Setup

The neural machine translation models were trained using Sockeye[7] (Hieber et al., 2017). We used 512-dimensional word embeddings and one LSTM layer in both encoder and decoder. We used word-count based batching (4096). Our systems employed the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of either 0.0002 or 0.0008 (see Section 3.5). The *dev* set from the corpus was used as a validation set for early stopping.

The baseline is an NMT model with the structure and hyperparameters described above without a curriculum; that is, it has access to the entire training set which

[7] `github.com/awslabs/sockeye`

is bucketed by length to then create mini-batches. Training data are split randomly into the same number of shards as the curriculum models (5 here).

We build the auxiliary model for the use of generating *one-best score* for each training sample, with similar but simpler configurations compared to the baseline model, in terms of number of RNN hidden units (200 vs. 512). While the training time for this specific model may cancel out the time saved by curriculum learning in practice, having a high-quality *one-best score* provides a useful reference point for our understanding of curriculum learning.

3.4.3 Curriculum Learning Setup

The curriculum learning framework as described in Section 3.3 was implemented within Sockeye. Curriculum learning can be enabled as an alternative to default training within Sockeye by specifying a file which contains sentence level scores (difficulty ranking per sentence with respect to any criterion). This implementation leverages the Sockeye sharding feature, which was originally meant for data parallelism. The codebase is publicly available with our experimental settings and tutorials[8].

We set the curriculum's update frequency to 1000 batches, which is the same as our checkpoint frequency.

[8] `https://github.com/kevinduh/sockeye-recipes/tree/master/egs/curriculum`

3.5 Results

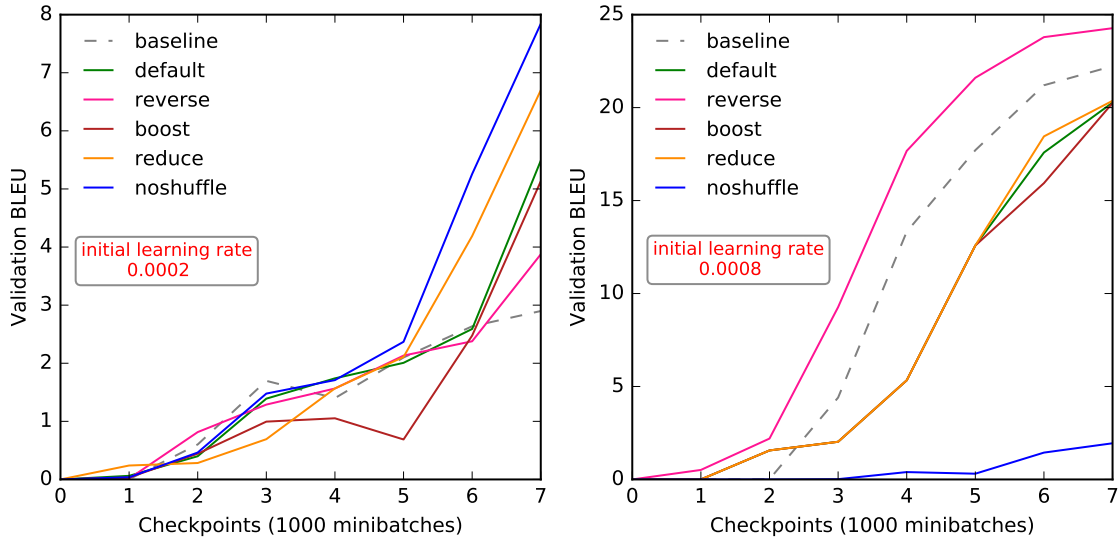


Figure 3.5: Learning curves for the first 7 curriculum updates. The NMT model is trained on data organized by the *avg word freq rank (de)* difficulty criterion with different curriculum learning schedules.

We start by examining training behavior during early training stages. Figure 3.5 shows the learning curves (validation BLEU[9] vs checkpoints) for the first 7 checkpoints[10] of curriculum training. The curriculum is updated at each checkpoint using one of the schedules listed in section 3.3.2. With the smaller learning rate, all curricula improve over baseline validation BLEU at the 7th checkpoint. However, with the higher learning rate, only the *reverse* schedule outperforms the baseline. Similar trends are observed with other difficulty criteria:[11] a few curriculum schedules beat

[9]BLEU is the standard evaluation for machine translation based on n-gram precision; higher is better (Papineni et al., 2002).

[10]7 is the lowest number of checkpoints required to discriminate between the different schedules.

[11]All learning curves available in the appendix (Chapter 8)

the baseline but this outcome is sensitive to the initial learning rate.

	Training Time (thousand batches)				
baseline	73				
	default	*reverse*	*boost*	*reduce*	*noshuffle*
one-best score	56	80	**59**	64	92
max wd freq(de)	57	88	89	82	77
max wd freq(en)	**63**	77	75	64	98
max wd freq(de-en)	**56**	61	**62**	**59**	**62**
ave wd freq(de)	**72**	**69**	57	73	108
ave wd freq(en)	84	66	61	61	**64**
ave wd freq(de-en)	62	57	84	85	**67**
sent len(de)	78	118	67	56	83
sent len(en)	151	59	67	125	196
sent len(de-en)	113	189	79	68	195
	Test BLEU (best)				
baseline	28.1				
	default	*reverse*	*boost*	*reduce*	*noshuffle*
one-best score	27.0	27.9	**28.4**	27.3	27.4
max wd freq(de)	25.2	26.1	27.4	27.2	28.1
max wd freq(en)	**27.6**	25.3	27.5	26.9	27.6
max wd freq(de-en)	**28.1**	27.5	**27.8**	**27.7**	**28.5**
ave wd freq(de)	**28.2**	**28.5**	27.3	26.5	28.2
ave wd freq(en)	27.8	25.4	27.4	25.8	**27.9**
ave wd freq(de-en)	27.3	27.4	28.3	26.9	**28.2**
sent len(de)	26.6	28.1	27.2	26.4	27.6
sent len(en)	27.6	25.1	25.6	27.1	27.7
sent len(de-en)	27.0	26.3	26.3	23.9	27.7

Table 3.1: Performance of curriculum learning strategies with initial learning rate 0.0002. Training time is defined as in Table 3.3. Bold numbers indicate models that win on training time with comparable (difference is less or equal to 0.5) or better BLEU compared to the baseline.

When training until convergence (Tables 3.1-3.2), 20 of 100 curriculum strategies successfully converge earlier than the baseline without loss in BLEU. The model trained with the average source word frequency as a difficulty criterion and the *reverse* schedule improves training time by 19% to 30%.[12] However, the optimal curriculum

[12]These are substantial time savings given that training the baseline took up to 1 day.

	Training Time (thousand batches)				
baseline	79				
	default	*reverse*	*boost*	*reduce*	*noshuffle*
one-best score	**59**	**69**	48	92	112
max wd freq (de)	85	103	**69**	118	**43**
max wd freq (en)	148	80	166	49	158
max wd freq (de-en)	84	**61**	**75**	67	93
ave wd freq (de)	79	**51**	73	88	58
ave wd freq (en)	**72**	71	146	61	74
ave wd freq (de-en)	81	47	54	58	71
sent length (de)	49	126	88	85	**74**
sent length (en)	101	52	70	49	114
sent length (de-en)	155	148	170	95	86
	Test BLEU (best)				
baseline	29.95				
	default	*reverse*	*boost*	*reduce*	*noshuffle*
one-best score	**30.1**	**29.9**	28.3	28.9	30.4
max wd freq (de)	25.9	29.6	**30.7**	25.8	**29.6**
max wd freq (en)	27.0	29.6	28.4	**29.5**	29.9
max wd freq (de-en)	29.5	**31.5**	**31.1**	27.9	27.2
ave wd freq (de)	27.3	**30**	27.6	27.1	21.3
ave wd freq (en)	**29.9**	28.4	23.3	25.2	29.4
ave wd freq (de-en)	29.9	28.4	28.5	28.3	29.3
sent length (de)	27.0	30.3	29.3	27.8	**31.0**
sent length (en)	29.0	27.6	24.2	26.9	30.2
sent length (de-en)	29.4	30.7	30.5	29.6	29.5

Table 3.2: Performance of curriculum learning strategies with initial learning rate 0.0008.

schedule for other complexity criteria change with the initial learning rate. The model trained with the *one-best score* and the *boost* schedule converges after processing 19% fewer mini-batches than the baseline (59,000 vs. 73,000) and yields a comparable BLEU score (28.4 vs. 28.1) with an initial learning rate of 0.002. With a higher initial learning rate, this configuration also speeds up training by 38% (48,000 vs. 79,000) but at the cost of a 1.65 point degradation in BLEU. The default schedule yields better results with the learning rate of 0.0008 but not 0.0002.

Comparing trends across complexity criteria shows there is no clear benefit to the expensive one-best model score compared to the simpler word frequency criteria. Sentence length is not a useful criterion: it helps convergence time only slightly (74,000 vs. 79,000) and in only one of the ten configurations we run. This is a surprising result at first, given that both sentence length and word frequencies were found to be useful ordering criteria by D. Zhang et al. (2017). However, their experiments are not directly comparable. They were limited to a single training epoch and use a different training strategy, which is closest to our *noshuffle* schedule. With that schedule, the German-English (de-en) sentence length curricula also outperform the baseline in early training stages, but the baseline catches up and outperforms by convergence time. We also note that the conclusions about the *reduce* stated by D. Zhang et al. (2017) do not hold true for our dataset and curriculum schedules. Specifically, this schedule provides no improvement in training time. (Table 3.1 and 3.2).

These results highlight the benefits of viewing curriculum learning broadly, and of curriculum strategies beyond the initial "easy samples first" hypothesis. Interestingly, the *default* and *reverse* schedules can yield close performance, and forcing data shards to be explored in order (*noshuffle*) does not improve over the *default* sampling schedule.

Table 3.3 further illustrates how curriculum training in NMT is sensitive to hyperparameters. We change the curriculum update frequency (mini-batches) and notice that while the validation set BLEU ramps up quickly, from 8.8 to 14.9, as the number

Curr Update Freq	**Time** (thousand batches)	**BLEU** (7)	**BLEU** (best)
1000	108	8.8	28.2
2000	100	1.8	28.0
3000	71	9.2	28.2
4000	56	9.0	27.9
5000	108	14.9	28.0
6000	67	14.9	28.0

Table 3.3: Impact of curriculum update frequency on the model trained on *default* schedule with data organized by *avg word freq rank (de)*. Training time is quantified as total number of mini-batches the NMT model has processed before convergence. The initial learning rate is set to 0.0002. The last two columns show the decoding performance of the model at 7th and the *best checkpoint* — the checkpoint at which the model got highest BLEU score on val set.

of mini-batches is increased between curriculum updates, the convergence time shows no clear trend and the validation BLEU at convergence is the same (ca. 28 BLEU).

To sum up, our experiments show that curriculum learning can improve convergence speed, but the choice of difficulty criteria is key: vocabulary frequency performs as well as the more expensive one-best score, and sentence length does not help beyond early training stages. No single curriculum schedule consistently outperforms the others, and results are sensitive to other hyperparameters such as initial learning rate and curriculum update frequency.

3.6 Synopsis

In this chapter, we conducted an empirical exploration of curriculum learning for training neural machine translation systems and its impact on convergence speed and quality on a German-English TED translation task. We adopted a probabilistic view

of curriculum learning, implemented on top of a state-of-the-art NMT toolkit, in order to enable a flexible evaluation of the impact of various curricula design.

We turn once again to the questions posed at the beginning of the chapter and examine them in the context of the results provided by this exploration.

1. We examined several hand-designed curricula, including the popular easy-to-hard (*default*) and its opposite, hard-to-easy (*reverse*), strategies which involve repetition (*boost* and *reduce*). Our results demonstrate that curriculum learning can be an effective method for training expensive models like those in NMT, as 20 of the 100 curricula tried improved convergence speed at no loss in BLEU.

2. We explored proxies for sample difficulty derived from auxiliary models and linguistic difficulty (word frequency and sentence length). We conclude that the choice of this difficulty criterion is key. No single curriculum schedule consistently outperforms the others, and results are sensitive to other hyperparameters such as initial learning rate and curriculum update frequency.

We note that while establishing an upper bound on performance for this task (an oracle) may be required to calibrate these results, devising an oracle is not an easy task. A naive approach would require searching through all possible (an exponential number) of data orderings to find the optimal one. An alternative approach is to synthetically add noise to some samples in the training dataset so that we have a prior notion of which samples are useful and which are not[13]. We will explore

[13]This may not be strictly true, as some noisy samples may still be useful.

these methods for determining oracles in the following chapters. Building upon these conclusions, we attempt in the next chapter to learn an optimal curriculum for the chosen NMT task, and compare the learned curriculum to hand-designed curricula as baselines.

Chapter 4

Reinforcement Learning based Curriculum Optimization for Neural Machine Translation

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) examples that are likely to be noisy or out of domain. Controlling the curriculum — the order in which examples are presented to the system — as is done in fine-tuning (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016), where training occurs first on general data, and then on more valuable in-domain data, has shown significant improvements in NMT training. Learning a curriculum could generalize

data filtering and weighting by allowing examples to be visited multiple times or not at all; and they additionally have the potential to steer the training trajectory toward a better final model than might be attainable with a static attribute-weighting scheme.

As we saw in the Chapter 3, 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 chapter we present an approach to meta-learning an NMT curriculum. Starting from scratch, we attempt to match the performance of a state-of-the-art (non-trivial) *reference curriculum* proposed by W. Wang, Watanabe, et al. (2018a), in which training gradually focuses on increasingly cleaner data, as measured by an external scoring function. Inspired by J. Wu, Li, and W. Y. Wang (2018), we develop 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 attributes and work with arbitrary corpora. Portions of the work described below appear in G. Kumar, Foster, et al. (2019). We begin by introducing our choice of the reinforcement learning framework, deep Q-learning.

4.1 Background

Suppose we have an agent interacting with an environment which consists of finite number of states $s \in \{s_1, \cdots, s_m\}$. At each time step, t, receives information about the state the environment is in and then chooses to execute an action $a_t \in A$, where A is a finite set of actions $\{a^{(1)}, \cdots, a^{(m)}\}$. Based on this action, the agent receives a reward (feedback) from the environment, r_t. We wish to train the agent to maximize the cumulative rewards which this agent receives over a time horizon. A popular reinforcement learning framework to address this problem is Q-learning (Watkins and Dayan, 1992) which learns a Q-table, a data structure which maps states and actions to their expected rewards. Specifically, Q-learning estimates a Q-value (estimated optimal future reward) for each state, action pair. At the beginning of training, the

Q-values are set to zeroes, indicating that the model has no knowledge of transitions in the environment. The model then proceeds to update these Q-values through its interactions with the environment which yield the triples (s_t, a_t, r_t) for each time step, t. The choice of actions is dictated by an exploration strategy which balances *exploiting* the Q-values from the model to determine the next action and *exploration* which chooses the next action at random. After each time step, the Q-learning model changes its estimates of the Q-values using the Bellman equation (Bellman, 1957) which appears below.

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \lambda \max_a Q(s_{t+1,a})) \tag{4.1}$$

where s_t and a_t are the state and action at time step t, r_t is the reward received, α is the learning rate and λ is the discount factor which balances how much we should use immediate versus historical rewards.

Deep Q-learning (Mnih et al., 2015) extends this approach to continuous state representations and uses deep neural networks to model Q-values instead of using a Q-table. This model takes the state representation and input and provides Q-values as output for every available action. Training of this network is performed using the Bellman equation which appears above. Deep Q-learning typically also uses two structurally identical networks, the *target* and the *main* to model Q-values. The

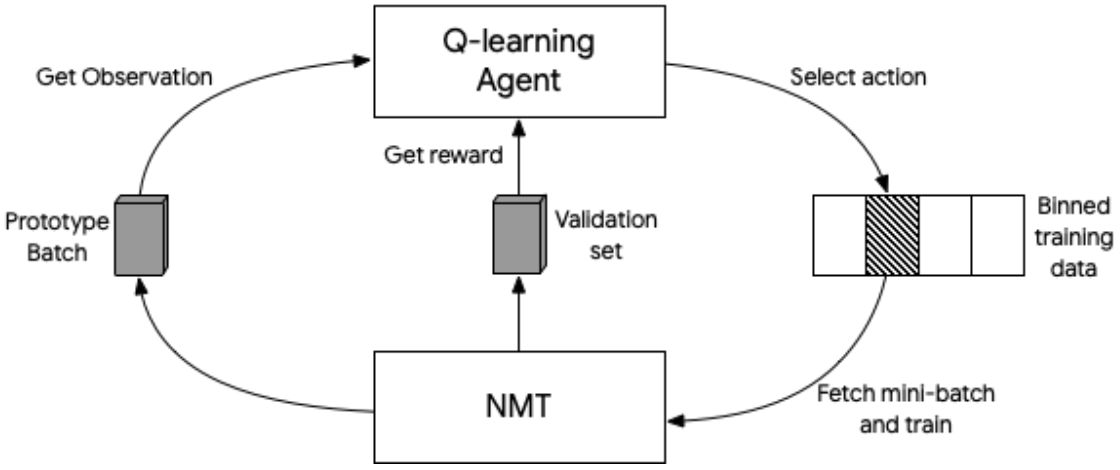


Figure 4.1: The agent's interface with the NMT system.

parameters of the target network are used to compute the estimated Q-value of the future state. The parameters of the *main* network are updated more frequently than the ones of the *target* network with the parameters from the former being copied to the latter every n updates (this is called the target update period). Additionally, deep Q-learning uses an *experience replay* which stores all action, state and reward triples. When a network needs to be trained, it samples a batch from this buffer to train and update its parameters.

4.2 Methods

4.2.1 Characterizing noise

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

W. Wang, Watanabe, et al. (2018a). 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 (clean) corpus, the score $s(e, f)$ for a translation pair e, f is defined as:

$$s(e, f) = \log p_{\theta_c}(f|e) - \log p_{\theta_n}(f|e) \tag{4.2}$$

Hence, CDS scores are essentially the conditional log-likelihood that an example comes from the clean data distribution. W. Wang, Watanabe, et al. (2018a) show that this heuristic 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.

4.2.2 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 illus-

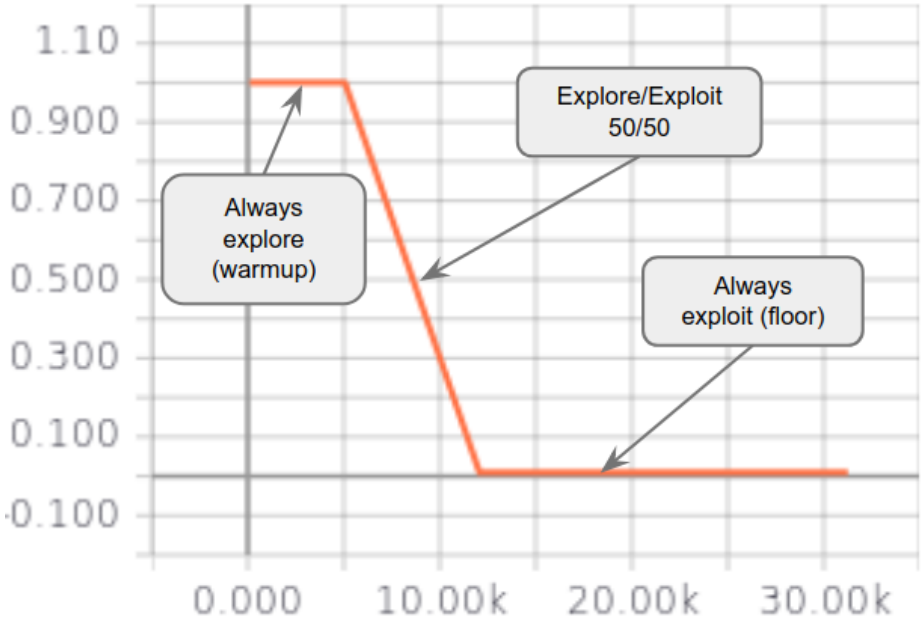


Figure 4.2: Linearly-decaying ϵ-greedy exploration.

trated in Figure 4.1. 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 update the NMT model.

4.2.3 Exploration Strategy

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 4.2). 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

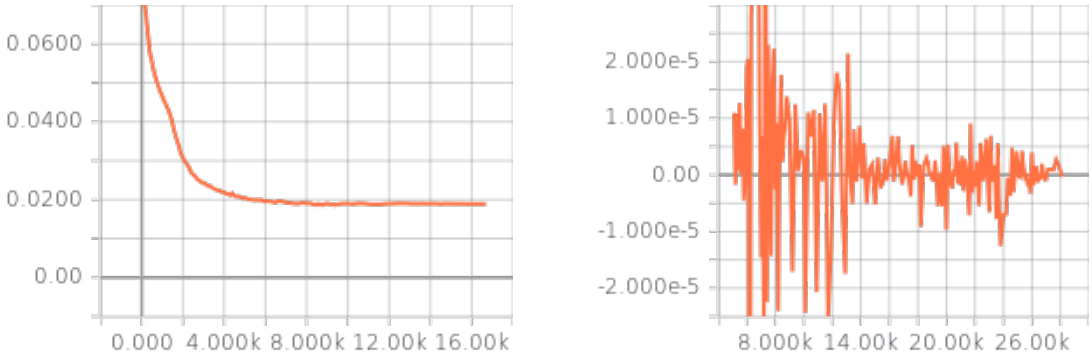
uninformed Q-function, the duration of exploration needs to be set carefully.

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.

4.2.4 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 J. Wu, Li, and W. Y. Wang (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 that 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.



(a) Perplexity (b) Delta-perplexity

Figure 4.3: Average rewards received by agent over time with the perplexity and delta-perplexity rewards.

4.2.5 Reward Engineering

Our objective is to find a curriculum which maximizes the likelihood of the NMT system under 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 (r) 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 NMT warmup period described in the previous section to eliminate spuriously large rewards at the beginning of training.

Specifically, this reward naturally decays over time as NMT training proceeds. A sub-optimal action by the agent may hence receive a larger reward simply by being

executed at the beginning of training. This may lead to optimal actions receiving diminished rewards later in training compared to sub-optimal actions receiving larger rewards at the beginning. To combat this, the actual reward function ($\hat{r}$) we use measures the improvement with respect to the average reward received in the recent past.

$$\hat{r}_t = r_t - \frac{1}{w} \sum_{i=t-w}^{t-1} r_i \tag{4.3}$$

where r_t is the actual reward received and $\hat{r}_t$ is the modified reward (delta-perplexity). Figure 4.3 shows the average reward received by agents over time with these two schemes.

4.3 Experimental Setup

Our NMT model is similar to RNMT+ (M. X. 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. An asynchronous evaluation process provides us with the log-likelihood of the development set with respect to the model parameters which determines our reward. With two evaluation processes running in parallel on two GPUs, we are able to obtain reward every ten steps on average although this frequency is not deterministic.

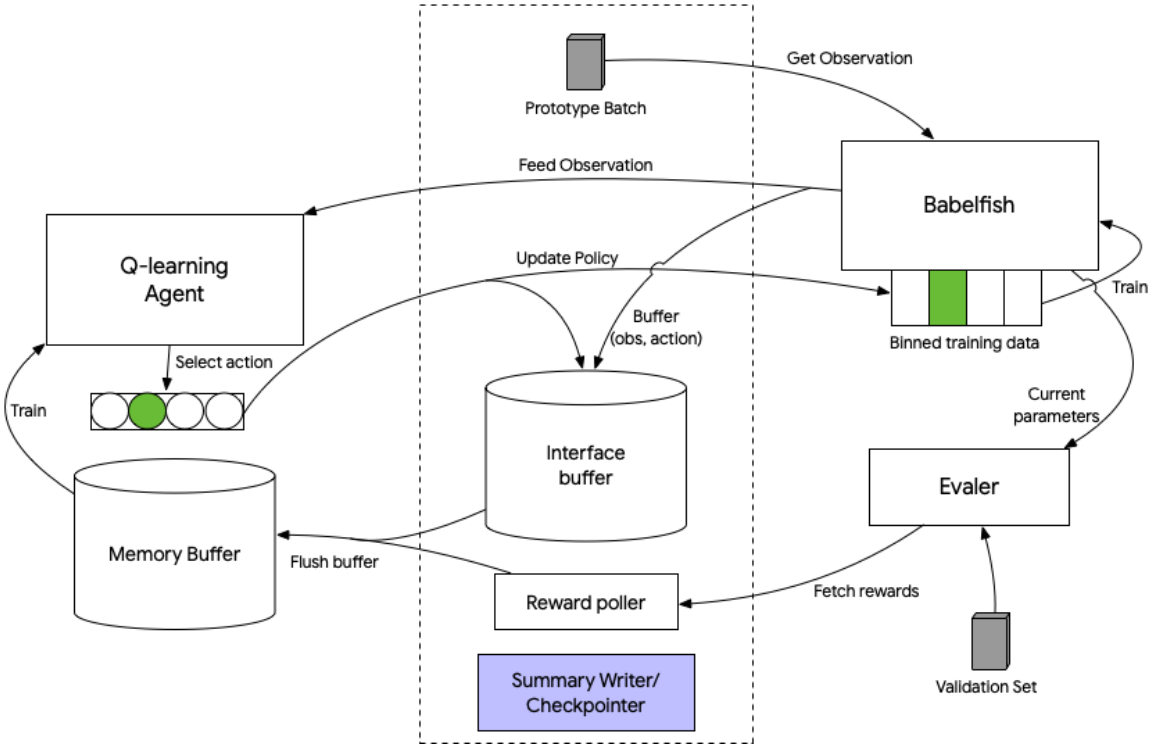


Figure 4.4: The RL agent's interface with the NMT system for Q-learning.

We use the DQN agent implementation in Dopamine,[1] which includes an experience replay buffer to remove temporal correlations from the observations, among other DQN best practices[2]. The RL-agent and NMT interface with the flow of control is shown in Figure 4.4 and is described next.

We begin by getting the observation by passing the *prototype batch* to the NMT system. The Q-learning agent uses this observation and its exploration strategy to determine the next action, the selection of a bin. A mini-batch is sampled from this bin and passed on to the NMT system for training. Due to the sparse and

[1] `github.com/google/dopamine`

[2] The Q-learning hyperparameters used in the experiments reported here are included in the appendix.

asynchronous nature of our rewards, we store observation, action transitions in a temporary buffer (*interface buffer* in Figure 4.4) 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 W. Wang, Watanabe, et al. (2018a), 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 W. Wang, Watanabe, et al. (2018a). 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

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

set was chosen. [3].

4.4 Results

Our results are presented in Table 4.1. **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
- *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 on Paracrawl

[3]The best RL-models were also, run an additional 2 times to verify the sanctity of the results.

English-French. 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 W. Wang, Watanabe, et al. (2018a) 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.[4] With 6-bins, it beats all uniform baselines by up to 2.5 BLEU and matches the hand-designed online baseline of W. Wang, Watanabe, et al. (2018a). On WMT, with 2 bins, the learned curriculum beats the 2-bin baseline, but not the uniform baseline over all data. With 6 bins, the learned curriculum beats the uniform baseline by 1.5 BLEU, and

[4]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 4.2: BLEU scores on ablation experiments with fixed rewards or observations on the Paracrawl En-Fr dataset.

matches the filtered baseline, which in this case outperforms the online curriculum by 0.6 BLEU.

Our exploration strategy for Q-learning (see Figure 4.2) 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 4.1) described above on both corpora makes this an unlikely explanation of its performance.

4.5 Analysis

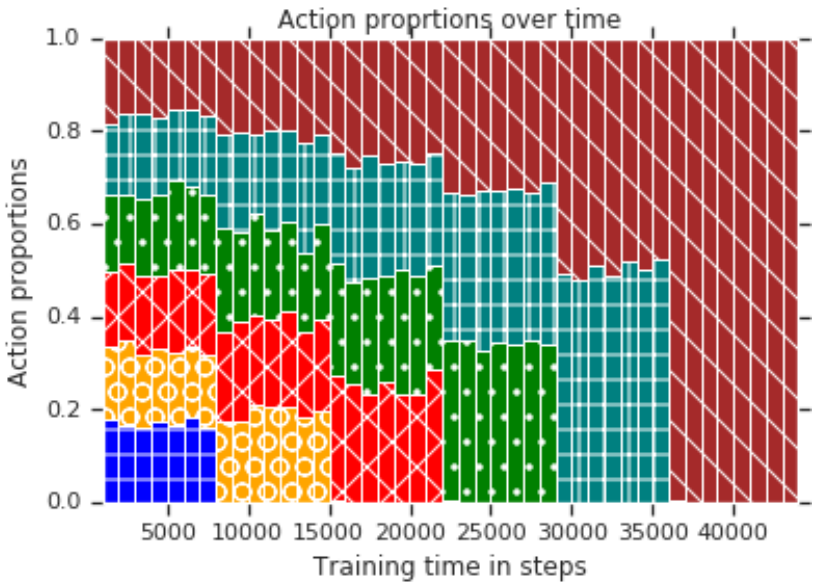
4.5.1 Information in observations and rewards

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

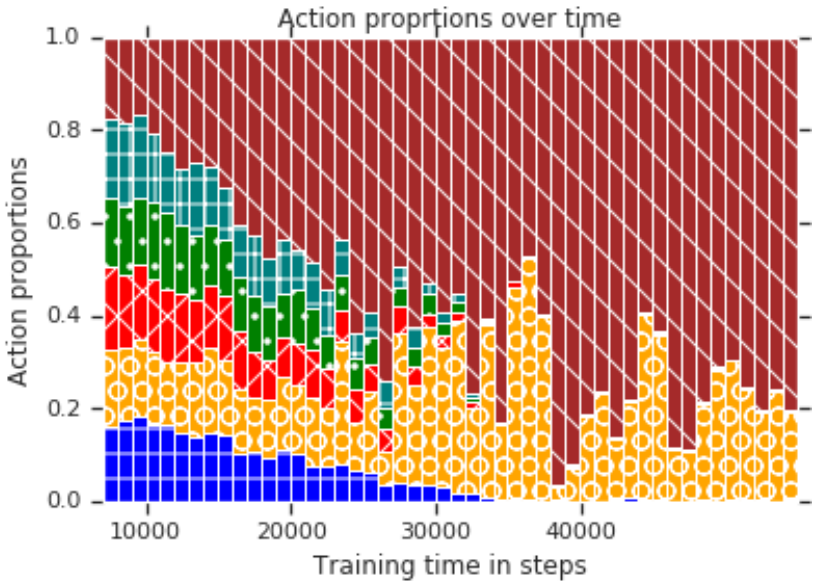
task. Table 4.2 shows the results of our experiments. The fixed reward experiments were conducted by replacing the default delta-perplexity based reward with a static reward which returns a reward of one when the cleanest bin was selected and zero otherwise. The fixed observation experiments used a static 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.

4.5.2 What did the agent learn?

Figures 4.5, 4.6 and 4.7 show a coarse visualization of the hand-optimized policy of W. Wang, Watanabe, et al. (2018a), adapted to our 6-bin scenario, compared to the Q-learning policy on the 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. The former (online), by design, telescopes towards the clean bins. Note that the latter (agent learned) policy is masked by the agent's exploration schedule, which slowly anneals

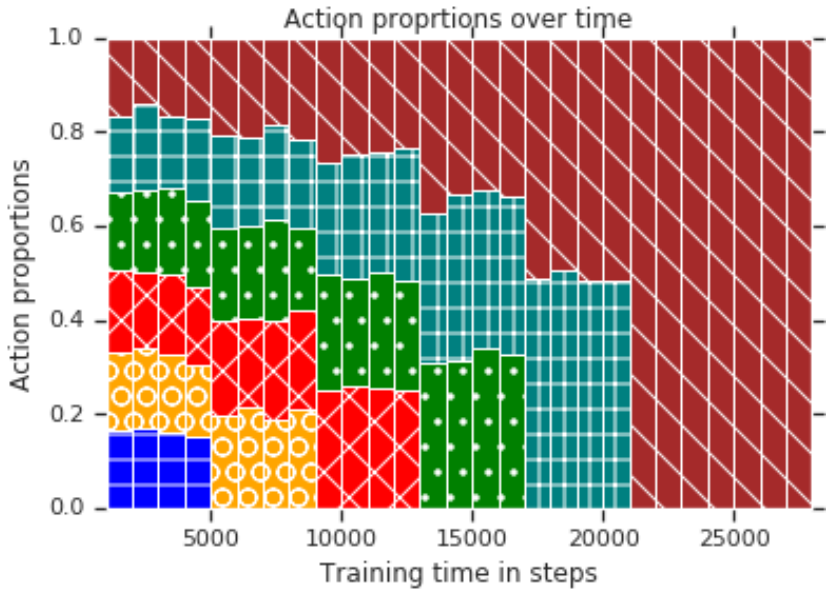


(a) Online

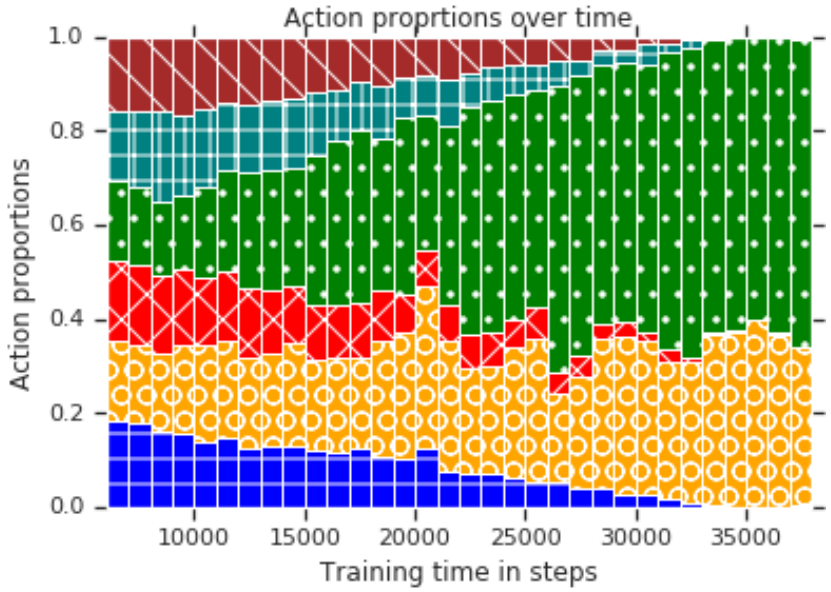


(b) RL learned

Figure 4.5: Online policy from W. Wang, Watanabe, et al. (2018a) 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.

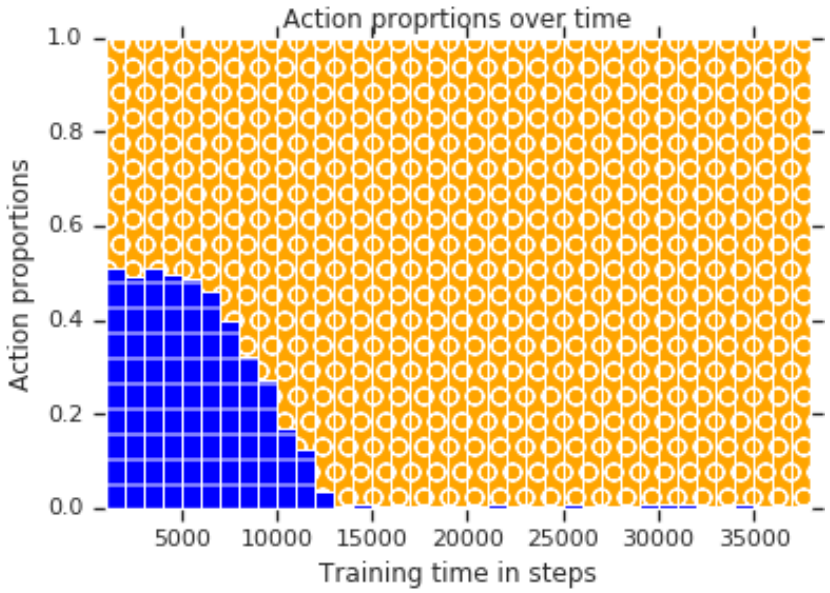


(a) Telescoping

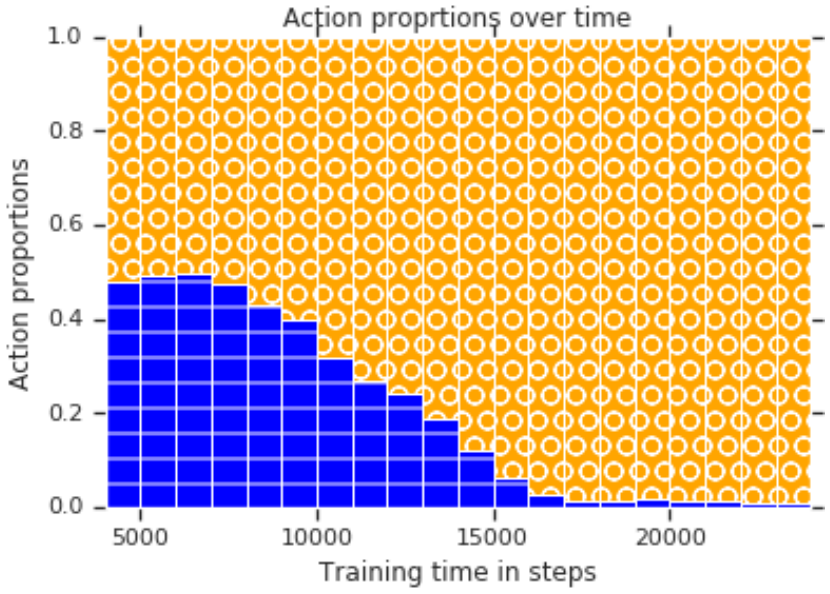


(b) RL learned

Figure 4.6: Policies learned by the RL agent on the WMT En-Fr corpus compared against the telescoping policy from W. Wang, Watanabe, et al. (2018a). Lower bins on the vertical axis contain more noise.



(a) RL Learned (Paracrawl)



(b) RL learned (WMT)

Figure 4.7: 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.

toward nearly complete policy control, beginning at step 30,000. After this point, the learned policy takes over and continues to evolve.

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 4.5) 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. We hypothesize that returning to the noisy bin acts as a form of regularization, though this requires further study[5].

3. The RL curriculum learned for WMT (Figure 4.6) 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 4.7 shows the policies learned on the bookend task for Paracrawl and WMT; the 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.

[5]As an example, we could force the model to train on only the cleanest bin once exploitation has started and compare against this experiment to verify this hypothesis.

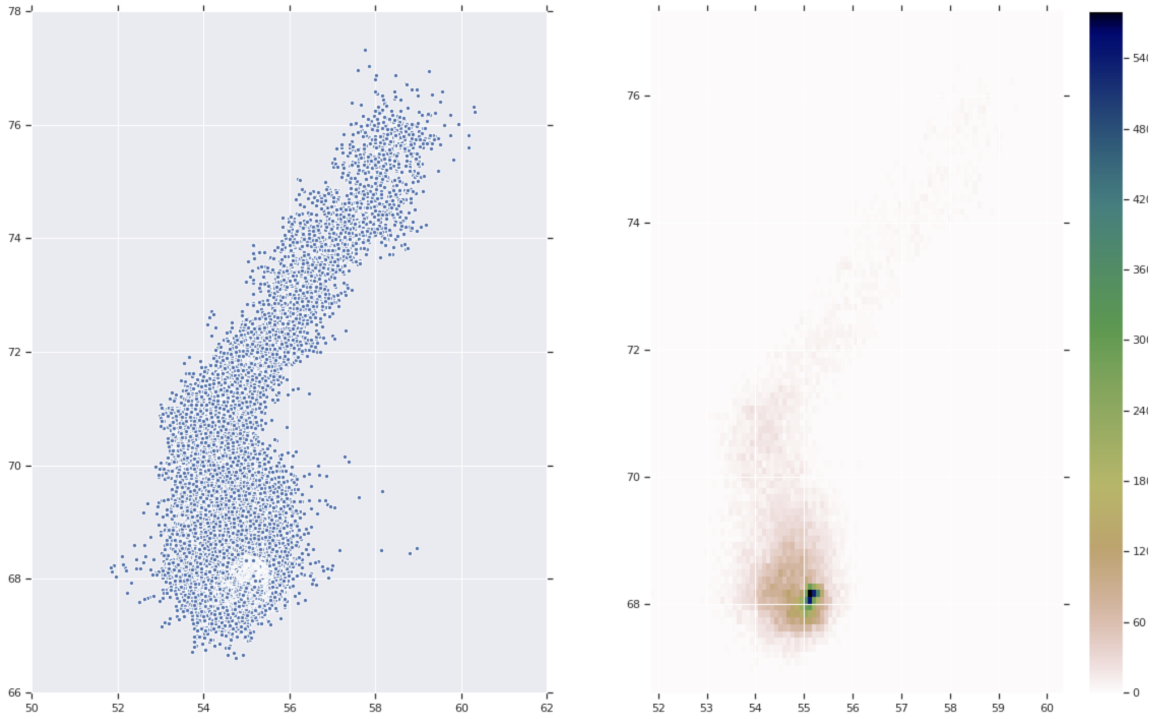


Figure 4.8: Sparsity of observations: Density of 2-bin mean of observations.

4.5.3 Sparsity of observations

A concern in trying to learn a policy from observations stemming from a single NMT run is that we may not receive enough samples in the space of possible observations. Figure 4.8 shows the density of the mean of observations in the 2-bin experiment on the Paracrawl dataset (Figure 4.7). The observations in this experiment settle into a fairly stable region after about 15000 steps. While it is interesting to ask if it is possible to significantly change the agent's belief in this stable region, an alternate approach which gathers observations from multiple training runs to cover more of the observation space may have greater success.

4.5.4 Dealing with instability

Q-learning and reinforcement learning in general are known to be unstable at times. While the changes proposed in Mnih et al. (2015) address some of these concerns, our learning scenario has unique problems which may make this problem worse. Specifically, the first time the agent is trained, the first batch (observation, action, reward) from the replay buffer determines a starting state for the agent. Given that exploration is expensive, an adversarial batch could put the best solution beyond reach. We suggest bootstrapping the agent by training it on multiple samples when its training begins to reduce the variance in the starting state.

4.6 Synopsis

In this chapter, we present a method which *learns* a curriculum for presenting training samples to an NMT system. This is in contrast to the approaches explored in Chapter 3 which were hand-designed; we use a state-of-the-art hand-designed curriculum as our baseline in this work. 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, the learned curriculum is able to match a state-of-the-art hand designed curriculum on Paracrawl and outperform it on the WMT dataset.

We see this a first step toward enabling NMT systems to manage their own training data. Two extensions to this work involve enabling the use of features other than noise (possibly multiple features) and dealing with the sparsity of observation problem (section 4.5.3). We tackle the former in Chapter 6 and address the latter in Chapter 5 by gathering observations from multiple training runs to learn more informed policies in the context of multi-lingual training.

Chapter 5

Learning Policies for Multilingual Training of Neural Machine Translation Systems

So far, we have explored hand-designed curricula and attempted to learn curricula jointly with training the NMT system. While both these approaches show improvements in convergence speed and/or translation quality, they come with their drawbacks. The former is hard to tune, relying on extensive trial and error to find the right hyperparameters, while the latter may suffer from observation sparsity, mainly because a single training run does not provide enough data sampling opportunities for an external agent to learn a good curriculum.

In this chapter, we build upon the framework for learning curricula established in

the previous chapter. To alleviate the problem of observation sparsity, we will attempt to learn more robust policies from multiple training runs. While we stay within the reinforcement learning framework to learn these policies, we will use contextual multi-arm bandits instead of Q-learning for our agents, since they are computationally less expensive to train. Additionally, we now add in some simple policy search methods to our list of baselines; specifically, we try and find the best policies using the expensive grid search and pruned-tree search methods. However, the state-of-the-art hand-designed curricula are still the baselines to beat.

In the previous chapter, we examined techniques for dealing with noise in datasets and while that is one typical characteristic of NMT training data, we would like to verify if these techniques work on other NMT tasks such as multilingual training, where the sample characteristic is the identity of the language-pair. Therefore, the NMT task of choice in this chapter will be *low-resource multi-lingual NMT* (MNMT). While standard NMT systems typically deal with one language pair, the source and the target, an MNMT model may have multiple languages as source and/or target. Most large-scale MNMT models are trained using some form of model parameter sharing (M. Johnson, Schuster, Quoc V. Le, et al., 2017b; Aharoni, M. Johnson, and Firat, 2019; Arivazhagan et al., 2019; Bapna and Firat, 2019). The notion of how input data should be presented to the MNMT system during training only finds prominence in the case of low-resource MNMT. A typical low-resource task will try to leverage a high-resource language pair to aid the training of an NMT system for

a low-resource (very small or no parallel data available) and related language-pair of interest. Typical approaches for low resource MNMT involve pivoting and zero-shot training (Lakew et al., 2018; M. Johnson, Schuster, Quoc V. Le, et al., 2017b) and transfer learning via fine-tuning (Zoph et al., 2016; Dabre, Fujita, and Chu, 2019). Finn, Abbeel, and Levine (2017) attempt to meta-learn parameter initialization for low-resource models using auxiliary-high resource models for this task.

Building upon the task and datasets established by Guzmán et al. (2019), we will attempt to learn a curriculum to train an NMT system for Nepali-English translation while leveraging the high resource Hindi-English pair. The agent will learn to choose between mini-batches containing either Hindi-English or Nepali-English data at each time step during NMT training to maximize the expected reward (improvement in validation set performance)[1]. The learned curriculum will hence condition on the state of the NMT system during training and determine whether to expose it to a batch of Nepali-English or Hindi-English data. Portions of the work described below appear in G. Kumar, Koehn, and Khudanpur (2021b). We begin with an introduction to contextual multi-arm bandits and our methodolgy for training them.

[1]Note that while this scenario may seem similar to the bookends task from the previous chapter, the latter contained one subset of data which was clearly sub-optimal for training, while in this case, both bins (Nepali-English) and (Hindi-English) have samples which, when trained on, can improve translation performance for the Nepali-Hindi language pair.

5.1 Background

Let us assume that we have an agent operating upon an environment. At each time step, t, the agent receives a representation of the state of the environment, the observation o_t, and must choose to take some action $a_t \in A$ from a set of possible actions $\{a^{(1)}, \cdots, a^{(m)}\}$. Based on the action which was chosen by the agent and then executed on the environment, we receive a real-valued reward, r_t. Note that the agent does not receive a reward for actions which it did not take. Our goal is to maximize the cumulative reward over time over a possibly infinite time horizon.

Contextual multi-arm bandits (Pandey et al., 2007; Chih-Chun Wang, Kulkarni, and Poor, 2005; Langford and T. Zhang, 2008), build a model for $P(r|a, o)$ from the observed (o, a, r) triples based on its interaction with the environment. As mentioned before, since the agent only receives rewards for the actions which it takes, and since at the beginning of training, the agent does not have a good model for $P(r|a, o)$, we need to use an exploration-strategy. The two commonly used strategies, epsilon-greedy (Tokic and Palm, 2011) and Thompson sampling (W. R. Thompson, 1933), balance exploration (choosing random actions) and exploitation (using the learned policy) to address these problems.

These contextual bandits are typically trained using the EXP3 (Auer et al., 2003) algorithm which maximizes the cumulative reward received by the agent, while minimizing regret (the difference in reward for the chosen action versus the best action for a time step; this is also called *weak regret*. An alternate version of training these

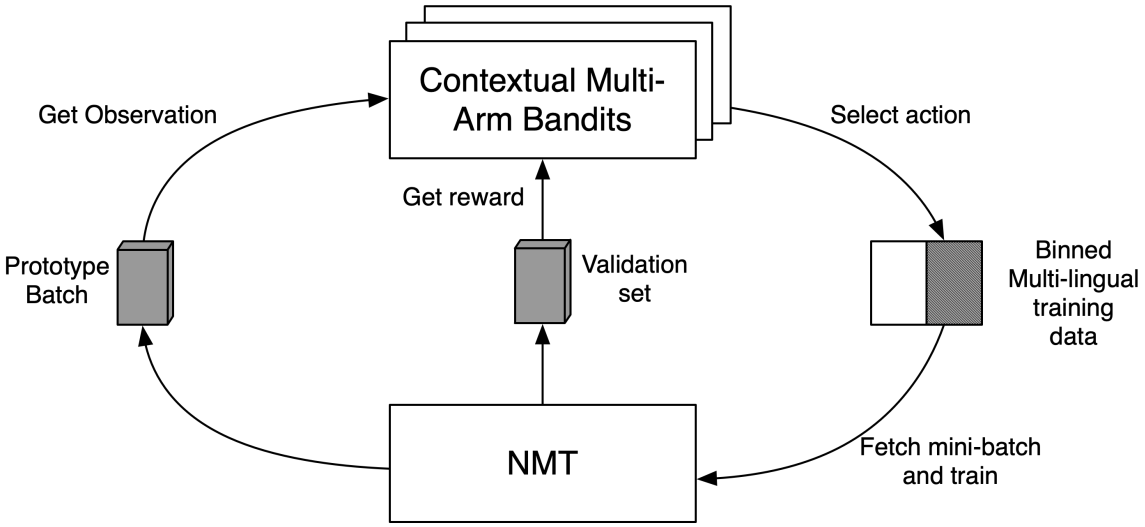


Figure 5.1: The multi-arm bandit agents' (MAB) interface with the NMT system.

bandits (and the one we use for this work) appears in Collier and Llorens (2018). They use multi-layer feed-forward networks to model $P(r|a, o)$ and choose an action (based on the exploration policy) which has maximum reward under this model. In our work, at each time step, the model is trained on a new batch of training data from its historical interaction with the environment.

5.2 Methods

5.2.1 Overview

The procedure for learning a two-bin policy for multi-lingual training will be similar to the one introduced in the last chapter. However, instead of using a single agent's experience in interacting with the *environment* (the NMT system), we will use

multiple multi-arm bandits which explore independent of each other and effectively learn their own policies. The stochastic nature of the exploration policy will ensure that they explore different spaces in the observation-reward space. This is done to address the observation sparsity issue which comes up in Q-learning (Mnih et al., 2015). Figure 5.1 shows an overview of this interface. The training data for all agents is pooled at the end of the training of individual agents and one final agent is trained using this data which determines the final policy we use as our multi-lingual curriculum.

5.2.2 Data Binning

Instead of mixing together all the language pairs into one single dataset, we create separate batches for each language pair. Hence, with respect to the agent, this is a two bin problem, where its *action* is the choice of the bin to draw a mini-batch. As a result of this design decision, each batch will only contain a single language pair. More generally, this can be extended to an arbitrary number of bins, one per language-pair being used to train the MNMT system.

5.2.3 Observation Engineering

The observations provided to the multi-arm bandits are identical to the ones introduced in the previous chapter (Section 4.2.4). A *prototype* batch is sampled per

bin (language-pair) and concatenated together. At each time step, the observation is the vector containing sentence-level log-likelihoods produced by the NMT system for this prototype batch. We exclude observations from the initial portion of NMT interaction to counteract the naturally decaying property of log-likelihood scores during NMT training.

5.2.4 Grid-search baselines

The simplest (albeit expensive to find) search-based learn-able curriculum to consider in this case is one where we sample batches from one language with a fixed probability or else sample from the other bin during training. Since there is only one degree of freedom in this search problem, we perform a simple line-search over the range of possible values for this probability. Note that, although this curriculum is 'learned' it remains fixed during training and does not change based on the state of the NMT system.

5.2.5 Pruned Tree search

A variation of the previous search method involves one which uses a technique similar to beam search. We divide training into a finite number of *phases* and then starting from the beginning of training, we search for the best fixed sampling probability. At the end of this *phase*, we discard all but the best model and the policy

Algorithm 1: Pruned tree-search for multi-lingual curricula search

Result: P^*, the list of the best policies per phase
$\hat{p} = \{0.0, 0.1, \cdots, 1.0\}$ // Policies to explore;
Randomly initialize starting NMT model Θ^*;
while *NMT next training phase t exists* **do**
  for p *in* $\hat{p}$ **do**
    Bin sampling probablity = p;
    Training start checkpoint = Θ^*;
    Run training of NMT training for phase t;
    Store trained model checkpoint θ
  end
  Select model θ^* with best score on validation set with policy p^*;
  $P^* = P^* + [(t, p^*)]$;
  $\Theta^* = \theta^*$;
end

which led to it, and continue the search for the best policy in the next phase from this model checkpoint. The result is a tree-search which prunes all but the best node after each phase. The final policy is the culmination of all phase-wise best fixed sampling ratios. This procedure appears in Algorithm 1. Note that a non-pruned version of this tree-search may yield an oracle sequence of bin selections (best bin at each phase) for the assumption that the best bin only changes at the end of the phase. However, this would require a search over an exponential number of nodes in the search tree. Hence, in a manner similar to beam search, we approximate this with this pruned search tree.

5.2.6 Contextual Multi-arm Bandits

Multi-arm bandit (MAB) based agents are typically trained to learn policies which maximize the expected reward received (minimize regret). Contextual multi-arm bandits (Pandey et al., 2007; Chih-Chun Wang, Kulkarni, and Poor, 2005; Langford and T. Zhang, 2008) allows the use of state based information to determine this policy. In our case the contextual MABs condition on the *observation* received from the NMT system to determine an *action*, the choice of bin to sample a mini-batch. The *reward* obtained for this action is the *delta-validation perplexity* post update as described in the previous chapter (Section 4.2.5). The exploration strategy is the linearly-decaying *epsilon-greedy* one, described in Section 4.2.3. The contextual MABs are implemented as simple feed-forward neural networks which take the *observation* vector as input and produce a distribution over two states representing the bins. If we choose to exploit this learned policy, the bin with maximum expected reward is selected for sampling.

5.3 Experiment Setup

We use Fairseq (Ott et al., 2019) for all the NMT experiments and the our NMT systems are configured to replicate the setup described in Guzmán et al. (2019). The grid search experiments search over the the range $[0, 1]$ for sampling in increments of 0.1. The pruned tree-search uses a beam width of 1. The phase duration for

Dataset	Sentences	Tokens
Nepali-English	563K	6.8M
Hindi-English	1.6M	16.7M

Table 5.1: Statistics of the training data for the Nepali-Hindi-English multilingual NMT system.

tree-search is set to one epoch of NMT training. We use either 5 or 10 concurrent contextual MABs which are implemented as two 256-dimensional feed forward neural networks. Further details about the hyperparameters for the contextual MABs are available in the appendix (Chapter 8). Rewards for the agent (validation delta-perplexity) are provided every ten training steps[2].

We use the datasets provided as part of the FLORES task (Guzmán et al., 2019) for our experiments. The statistics of the training dataset for the multi-lingual task appear in table 5.1. The Hindi-English dataset comes from the IIT Bombay corpus[3]. The validation and test sets for Nepali-English (the low resource language-pair of interest) contain 2500 and 3000 sentences respectively.

5.4 Results

Our results are presented in Table 5.2. Our baselines consist of:

- ne-en random baseline: This is the NMT setup which is only trained on the

[2]As an example, if the current policy dictates that the sampling probability for Nepali-English is 0.2 (and for Hindi-English is 0.8) then, the NMT model is updated 10 times using batches sampled based on these probabilities. The resulting model is then used to evaluate the validation set to compute the reward.

[3]`http://www.cfilt.iitb.ac.in/iitb_parallel`

	valid	test
Baselines		
ne-en: Random Baseline	6.35	7.71
hi-en: Random baseline (with ne valid)	2.71	3.9
ne-hi-en: Random Baseline	**12.24**	**14.88**
ne-hi-en: Multi-lingual Transformer	12.01	14.78
ne-hi-en: Continued training from hi-en	12.2	14.3
Searched Curricula		
Grid Search (best = 50/50)	12.01	14.78
Grid Search (best = 50/50) + Continued training	**12.33**	**15.1**
Pruned Tree-search	12.3	14.8
Pruned Tree-search + Continued training	12.41	14.92
Agent Learned Curricula		
MAB1 (best = 10 concurrent, 500 updates)	**12.21**	**14.87**
MAB2 (best = 5 concurrent, 2 epochs)	12.18	14.67

Table 5.2: BLEU scores for the Nepali-English test set using the fixed, searched and learned multilingual curricula.

Nepali-English corpus. The data is randomly shuffled to form mini-batches.

- hi-en random baseline: The NMT system trained on the high-resource Hindi-English dataset with the Nepali-English validation and test sets.

- ne-hi-en random baseline: The Hindi-English and Nepali-English data is mixed together to train the NMT system. The Nepali-English data is upsampled to match the size of the the Hindi-English corpus. This is roughly identical to sampling the Hindi-English and Nepali-English corpora with the same probability except for the fact that the resulting batches in this case are not homogeneous with respect to language-pair identity.

- Multilingual transformer: Replicates the setup from Guzmán et al. (2019).

- Continued training baseline: Uses the hi-en random baseline as a starting point and continues to fine tune on the Nepali-English training data using the Nepali-English validation and test sets.

Our non-MAB search-based curriculum baselines are:

- Grid search: A static curriculum is learned by searching over the space of sampling probabilities for the bins.
- Grid Search + Continued training: The previous model is fine tuned using the Nepali-English validation and test sets.
- Pruned tree-search: Epoch-dependent curriculum searched using the pruned tree-search method.
- Pruned tree-search + Continued training: The previous model is fine tuned using the Nepali-English validation and test sets.

From Table 5.2, we see that the ne-en and hi-en baselines are very weak, with the latter lagging behind despite having access to more data. This indicates that with these language pairs, even though using the high-resource dataset may be worth something, if no low-resource data is available, it is not a good proxy for the low-resource pair. The random baseline with the combination of the two datasets (upsampled low-resource) is the strongest amongst the fixed baselines marginally beating the multilingual transformer and the (surprisingly) the continued training baselines. While

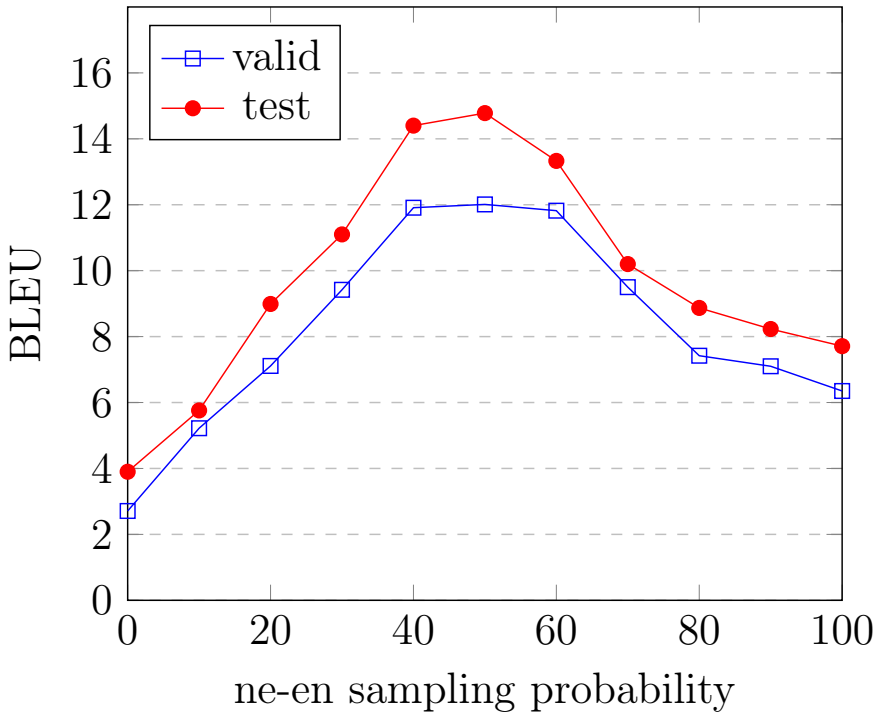


Figure 5.2: BLEU scores for the Nepali-English validation and test set at various values of the ne-en sampling probability.

	valid	test
MAB (5 concurrent, 500 updates)	12.2	14.11
MAB (10 concurrent, 500 updates)	**12.21**	**14.87**
MAB (5 concurrent, 1 epoch)	11.44	13.98
MAB (5 concurrent, 2 epoch)	12.18	14.67
MAB (10 concurrent, 500 updates) + sampled valid reward	12.18	14.23
MAB (5 concurrent, 2 epoch) + sampled valid reward	12.21	14.5

Table 5.3: BLEU scores for the Nepali-English test set using various configurations of the contextual MABs to learn the multilingual sampling curriculum.

the grid search and pruned-tree search baselines are close in performance to the best fixed baselines, continued training with them provides much stronger results where the 50/50 configuration for the grid search provides the best result at 15.1 BLEU and the tree search slightly behind at 14.92 BLEU. Figure 5.2 shows the BLEU scores for the grid search experiments over the chosen search points in the probability space.

For the contextual MABs, we use either 5 or 10 concurrent agents (training data is gathered from all concurrent bandits to train the final curriculum). In addition, we choose to update the bandit policy only once every 500 updates, 1 epoch or 2 epochs of NMT training. The results of all our experiments appear in table 5.3 and the best configurations are in table 5.2. While the curricula learned using the contextual MABs are able to match the performance of the strongest fixed policy (ne-hi-en random baseline), it performs slightly worse than the curriculum obtained using the (expensive) grid search combined with continued training, by about 0.2 BLEU points.

5.5 Synopsis

In this chapter, we build upon the approach presented in Chapter 4 by learning from multiple NMT training runs. On the task of low-resource multilingual NMT training, we learn a curriculum using conditional multi-arm bandits which conditions on the state of the NMT system and decides to either train on a batch of a high-resource (Hindi-English) or the low-resource (Nepali-Hindi) language pair. In addition, we introduce some simple search-based methods for policy search (grid search and pruned tree search) for this task. We show that both these simple *learned* curricula and the ones derived from the MABs can match the state-of-the-art hand-designed multilingual baselines. However, continued training with these *learned* curricula provide slightly better results, indicating that they may serve as good starting models for fine-tuning (another possible benefit of curriculum learning).

Additionally, as discussed in Chapter 4, reinforcement learning based deep learning techniques tend to be unstable (Mnih et al., 2015) when (i) the training samples for agent learning are correlated[4] and (ii) if not enough samples are available. We address both these concerns in this chapter by accumulating training samples for the contextual multi-arm bandits from multiple NMT training runs and then shuffling them to achieve decorrelation. We also use a simpler reinforcement learning framework (contextual MABs vs Q-learning) for this task. These changes implemented

[4]We address this in Q-learning by using a *replay buffer*, an accumulator for the training samples, from which samples are randomly (and not sequentially) sampled to train the agent.

together ensured that we observed no instability in agent-learning for this task.

In the previous chapters, we have used a single proxy for sample usefulness, linguistic difficulty, noise, auxiliary model score or language-id. While these have been useful, in the next chapter, we show that a single feature may not be sufficient to capture the notion of usefulness of a training sample to curriculum design. We propose methods which allow the use of multiple features allowing us to learn sample usefulness and the curriculum jointly with the training of the NMT system.

Chapter 6

Learning Feature Weight based Policies for Denoising Parallel Corpora

In the previous chapters, we have relied on a proxy feature to determine sample usefulness for training and in turn for curriculum design. In Chapter 3 we used linguistic features and auxiliary model scores, Chapter 4 used noise based CDS scores and Chapter 5 used language pair identity. These proxies for usefulness have shown promise in our previous work and are a convenient way to infuse prior knowledge into the design of a curriculum; e.g., we may hypothesize that noise based features are the most useful ones to use when training an NMT system of a very noisy corpus. However, we acknowledge that the notion of sample usefulness as determined by a

single feature is an artificial one and a hard problem in itself to tackle; it may not always be clear which single feature is a good proxy for a task. In this chapter, we present a method which does not require a single feature to be specified but instead can use multiple ones. The usefulness of a sample for the task, and the curriculum, are learned jointly with the training of the NMT system. Our task of choice will be denoising – learning a curriculum for filtering, where some samples are never presented for training based on the learned sample usefulness score – using large noisy parallel corpora.

Large parallel corpora such as Paracrawl (Bañón et al., 2020) which have been crawled from online resources hold the potential to drastically improve performance of neural machine translation systems across both low and high resource language pairs. However, since these extraction efforts mostly rely on automatic language identification and document/sentence alignment methods, the resulting corpora are extremely noisy. The most frequent noise types encountered are sentence alignment errors, wrong language in source or target, and untranslated sentences. As outlined by Khayrallah and Koehn (2018), training algorithms for neural machine translation systems are particularly vulnerable to these noise types. As such, these web-crawled corpora have seen limited use in training large NMT systems.

In this chapter, we propose a method for denoising and filtering noisy corpora that explores and searches over weighted combinations of features. During NMT training, we score sentences and create batches using random weight vectors. These

batches are used to train the system and measure improvement over the validation set (reward). Finally, by modeling the weight-*reward* function, we learn the set of weights which maximize reward and are used to score and filter the noisy dataset. At a high level, this method (i) allows the use of multiple sentence level features, (ii) learns a set of interpolation weights for the features which directly maximize translation performance, (iii) requires no prior knowledge about which features are informative or even if they are mutually redundant, and (iv) trains within the NMT pipeline and does not require any special infrastructure.

We include experiments which apply this learned filtering curriculum to building NMT systems for the noisy Estonian-English Paracrawl dataset and show that it beats strong single feature filtering-baselines and hand-designed feature interpolation. Additionally, we analyze the robustness of this method in the presence of specific kinds of noise (Khayrallah and Koehn, 2018) via a controlled experiment on the Europarl datasets. Finally, we look at the impact of transferring the learned weights from one language pair (Estonian-English) to a noisy dataset of another language pair (Maltese-English Paracrawl). The work presented in the chapter appears in G. Kumar, Koehn, and Khudanpur (2021a).

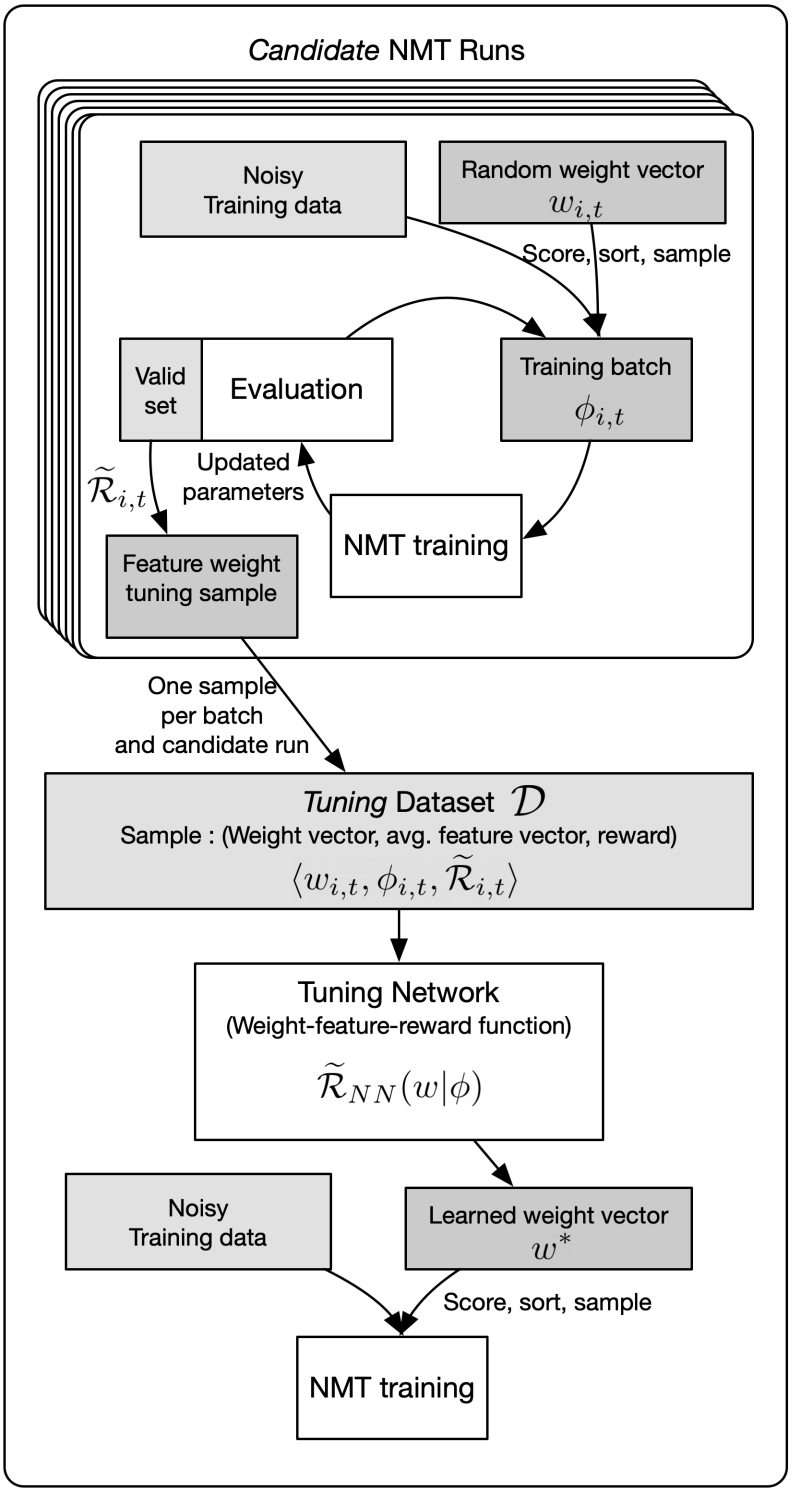


Figure 6.1: Overview of the proposed method for learning weights for sentence-level features to filter noisy parallel data and improve translation performance.

6.1 Methods

The proposed method centres around finding weights for combining sentence-level features, which are then used to compute sentence-level scores and filter the noisy corpus. While the choice of features can be arbitrary, this method's performance will eventually depend on their quality, and we would ideally want them to be informative and uncorrelated.

Figure 6.1 provides an overview of the proposed method. We first train a number of *candidate* neural machine translation (NMT) systems. During training for each candidate system, we repeatedly (i) generate a random weight vector, (ii) sample a batch of sentences from the noisy corpus based on sentence-level scores computed using this weight vector, (iii) update NMT system parameters using this batch, and (iv) measure the improvement in translation quality on a validation set following this update. The weight vector w, the average feature vector ϕ of sentence-pairs in the batch, and the improvement r on the validation set (*reward*) are recorded for each batch t during the training of each candidate NMT system i, and $\langle w_{i,t}, \phi_{i,t}, r_{i,t} \rangle$ becomes a sample in new data set $\mathcal{D}$, called the *tuning* data set[1], for learning feature weights to maximize reward. Hence, even though the translation model parameters of the *candidate* NMT systems are not used directly, they are used to gather noisy *candidate* evaluations of the latent weight-feature-reward function.

[1]Not to be confused with the validation set which contains sentence pairs, this dataset is solely used to model the weight-reward function and contains no sentence identity beyond feature vectors.

Once we have $\mathcal{D}$, we use a feed-forward network to learn the weight vector that maximizes the reward. The learned weight vector w^* is then used to compute sentence-level scores and filter the noisy data set. The *final* NMT system is trained using this *clean* data set.

Some subtleties in normalizing the observed rewards and learning weights are explained below.

6.1.1 Candidate NMT runs

Note from the bottom of Figure 6.1 that the learned weight vector w^* is used to sort all the sentences in the noisy training data, and the top-scoring ones are used for final NMT training. The purpose of the candidate NMT training runs is to generate the *tuning* data set $\mathcal{D}$ from which w^* is learned. Therefore, the setup for the candidate runs mimics typical NMT training, but with the following differences.

1. **Selecting batches**: For selecting sentences to constitute a batch, we first sample a random weight vector w of dimension $|\phi|$, the number of sentence-level features, uniformly[2] from $[-2.5, 2.5]^{|\phi|}$. Ideally, we would score *all* sentences in the noisy data set and then filter the top sentences to create a batch. However, this is prohibitively slow to do for every batch. As a shortcut, we randomly sample twice the number of sentences required to constitute the batch, score them using the same random vector, and select the top half. For the i^{th} sentence,

[2]The range of the uniform distribution represents the plausible range of weights given the features.

the score s_i is a dot product of its feature vectors with the weight vector:

$$s_i = \sum_{i=1}^{|\phi|} w_i \phi_i \tag{6.1}$$

The selected sentences are removed from the training pool for this epoch. This method of batch selection ensures that the sampled weight vector determines which sentences are selected and that their average feature vector is significantly different from one obtained using unbiased/random selection.

2. **Reward computation**: The reward must represent how the choice of w (through the sentences selected to form the batch) impacts translation performance. This is approximated by computing the perplexity of a validation set following a parameter update with the selected batch. However, since perplexity naturally decays in standard NMT training, batches at the beginning of the training will naturally receive larger rewards, obscuring the impact of sentence selection. We mitigate this effect by using delta-perplexity, i.e. the change in perplexity of the validation set over a window of updates.

3. **Accumulating training samples**: For each batch t of candidate run i, we collect the random weight vector $w_{i,t}$, the batch feature vector $\phi_{i.t}$, defined as the average of the feature vectors of all sentences in the batch, and the reward $r_{i,t}$. These triples are gathered from all batches during training, across all candidate training runs, to form the data set $\mathcal{D}$ for learning the feature weights.

6.1.2 Reward Normalization

As a further way to make the rewards time-invariant with respect to NMT training, the observed rewards $r_{i,t}$ are normalized with respect to an expected reward estimated from a set of *baseline* NMT runs. Specifically, at each time step t, we compute the rewards $r^b_{j,t}$ of $j = 1, \ldots, J$ concurrent training runs—whose batches selected in the standard manner—and, for each of the candidate NMT runs, we set

$$\widetilde{r}_{i,t} = r_{i,t} - \frac{1}{J} \sum_{j=1}^{J} r^b_{j,t} \tag{6.2}$$

where J is the number of baseline systems used.

Going forward, we do not need to track the identity of the update which led to a training sample, t, or the candidate system c_i which produced it. Note that this leads to the learning of a policy which is is time independent. That is, we are trying to learn a policy which is optimal for all time steps in training. While this is appropriate for the fixed filtering curriculum we are attempting to learn in this chapter, if we desire a time and state dependent policy, the approaches such as the ones presented in Chapter 4 and Chapter 5 may be more relevant.

6.1.3 Learning Feature Weights

The i^{th} sample $\langle w_i, \phi_i, \widetilde{r}_i \rangle$ in $\mathcal{D}$ may be viewed as a (noisy) evaluation of an unknown function $\mathcal{R}(w|\phi)$. This function maps a vector w to final NMT quality, given a

fixed sentence-level feature function ϕ and the stipulation that sentences are selected for training based on a weighted combination of their feature values using weights w. Furthermore, if we learn this function using $\mathcal{D}$, we may use the w^* that maximizes the learned function $\widetilde{\mathcal{R}}_{NN}(w|\phi)$ for our final denoising and NMT training. Specifically, we propose to use

$$\begin{aligned} w^* &= \arg\max_w \mathcal{R}(w|\phi) \\ &\approx \arg\max_w \widetilde{\mathcal{R}}_{NN}(w|\phi) \end{aligned} \tag{6.3}$$

We propose learning $\widetilde{\mathcal{R}}_{NN}(w|\phi)$ via a simple feed-forward neural network that maps the weights w_i to the observed reward $\widetilde{r}_i$. We consider two ways of providing input to this neural network, one that uses only the w_i, and another that modulates w_i with batch quality, represented by ϕ_i.

1. **Weight-based**: We use a feed-forward network with the weight vectors w_i as input and learn to predict the observed reward $\widetilde{r}_i$. Since the weight vectors interact directly with the feature vectors to determine which sentences are sampled to create a batch, we hypothesize that maximizing this weight-reward function will produce feature weights which will lead to better sentence sampling.

2. **Feature-based**: Since the *tuning* samples are noisy evaluations of the function $\mathcal{R}(w|\phi)$, we often encounter samples where weight vectors are close in weight space but have different rewards. To counter this problem, when using a feed-

forward network to learn $\widetilde{\mathcal{R}}_{NN}(w|\phi)$, we scale the weight vector input w_i by the sum of the corresponding feature vector ϕ_i. This has the effect of keeping weight vectors which have similar feature vectors close in input space and moving apart those with significantly different feature vectors.

Once this neural network is learned from $\mathcal{D}$, we perform a grid search over its input space, as defined in Section 6.1.1, to find the maximizer of (6.3).

6.1.4 Re-sampling and training

The weight vector w^* learned from the previous section is used to score all sentences from the original noisy data set. We sort the sentences by these scores and sample the top candidates to form the *clean* training data set and use it to train a standard NMT system.

6.2 Experiment Setup

We use Fairseq (Ott et al., 2019) for our neural machine translation systems configured to be identical to the systems described in Ng et al. (2019). The feed-forward network used to tune weights has two 512-dimensional layers and is trained using standard SGD using a learning rate of 0.1. The grid search for the weights was done on the range $[-2.5, 2.5]$ with 5000 points uniformly distributed per dimension. The number of samples used for reward normalization (J in eqn. 6.2) was 3 and the

window for computing the delta-perplexity reward was set to 3.

6.2.1 Corpora

We use the Paracrawl Benchmarks (Bañón et al., 2020) data set in Estonian-English for all our experiments. These consist of documents where sentences were aligned using Vecalign (B. Thompson and Koehn, 2019) and then de-duplicated so that each sentence pair only occurs once in the data set. The test and validation sets for our experiments in Estonian-English are *newstest2018* and *newsdev2018* respectively. Statistics of these corpora appear in Table 6.1.

	train	**valid**	**test**
Sentence Pairs	22.8m	2k	2k
Source Tokens	190m	29k	31k
Target Tokens	207m	38k	40k
Avg. Len (src)	9.8	14.5	15.3
Avg. Len (tgt)	10.7	19.1	20.1

Table 6.1: Statistics for the processed Estonian-Engligh (Es-En) Paracrawl data set and its corresponding validation and test sets. The training corpus was filtered using Vecalign scores; the raw corpus contains about 168m sentence pairs.

6.2.2 Features

We use five sentence-level features for all our filtering experiments. They are, (i) IBM Model 1 alignment scores (Brown et al., 1993), (ii and iii) source and target language model scores, (iv) dual conditional cross entropy (Junczys-Dowmunt, 2018) and (v) sentence length ratio. We experimented with aggregate features such as

Zipporah (Xu and Koehn, 2017), BiCleaner (Sánchez-Cartagena et al., 2018) and bilingual features such as LASER (Schwenk and Douze, 2017) and these were used to replicate the baselines from Bañón et al. (2020) for our dataset. The IBM Model 1 scores were obtained using the Moses (Koehn et al., 2007) pipeline. The Estonian and English language models were trained on their respective NewsCrawl data sets[3]. The *clean* machine translation model for computing the conditional dual-cross entropy scores is trained on the *Europarl*v8 data set[4]. All features are Gaussianized using the Yeo-Johnson (Yeo and R. Johnson, 2000) power transformation and then normalized to have zero mean and unit variance.

6.3 Results

For our experiments, we scored all sentences in the noisy corpus, sorted and sampled the top parallel sentences to form subsets with 10, 15 and 20 million English words. These filtered data sets were used to train standard NMT systems and performance was evaluated on the test set described in the previous section. The results of these filtering experiments appear in Table 6.2.

First, we evaluate the efficacy of all the features we use for our interpolation task by filtering the data set on these features alone. Additionally, to include some strong baselines, we use three out-of-the-box, scoring features which provided strong results

[3] `statmt.org/wmt18/translation-task.html`
[4] `statmt.org/europarl/`

in the WMT 2020 parallel corpus filtering task[5] (Bañón et al., 2020; Chaudhary et al., 2019). These are BiCleaner, Zipporah and LASER. Of these, LASER provides the strongest filtering and translation results beating the other two by 0.3 to 0.9 BLEU points. Of the five features we use for our experiments, dual cross-entropy (Junczys-Dowmunt, 2018) is the strongest feature and matches the performance of LASER. Using source or target language model scores in isolation leads to the weakest translation performance while IBM Model 1 scores perform only slightly better than them. Surprisingly, the simple sentence length ratio feature beats all other features except dual cross-entropy by 1.4 to 1.6 BLEU points. This is a strong indicator of the type of noise in the data set and that bilingual features (even simple ones) perform better than monolingual features such as language model scores.

Next, we look at interpolation of features using weights learned using the proposed method. As a baseline, we also include an experiment which filters based on a uniform interpolation of the five features we use. This baseline performs worse than the strongest single feature filtering experiments by 0.5 to 1 BLEU points. For both the weight-based and feature-based methods of learning interpolation weights for the features, a significant number of *candidate* runs are required before adequate performance is achieved. This is not surprising, since we are searching for an optimal weight vector in a fairly large weight space and we need a large number of samples before a good representation of the weight-reward function can be learned. Figure 6.2

[5] `statmt.org/wmt20/parallel-corpus-filtering.html`

	10m	**15m**	**20m**
1-Feature Filtering Baselines			
Zipporah	20.4	21.3	21.3
BiCleaner	19.8	20.9	21.2
LASER	21.7	22.4	22.5
IBM Model 1	18.1	19.9	20.8
Target LM	17.6	19.5	20.4
Source LM	17.4	19.4	20.4
Dual Cross-Entropy	21.5	22.4	22.6
Sentence Length Ratio	19.7	20.2	21.2
Filtering using Feature Weights			
Uniform weight baseline	20.9	21.5	21.6
Weight based (14)	22.1	**23.1**	**23.5**
Feature based (15)	**22.4**	**23.1**	**23.6**

Table 6.2: BLEU scores for the Estonian-English NMT systems where the training data was filtered using single features or a learned weighted combination of features. Feature weights were learned using the proposed method. The number of candidate runs which produced the best results appear in parentheses.

shows the improvement in BLEU scores for the weight-based approach as data from more *candidate* runs in added to the *tuning* stage for learning weights and filtering the data set. The performance of the final NMT system steadily improves as more data from more systems is added and eventually converges.

Our strongest result was achieved with 14 *candidate* runs for the weight-based approach for the 10, 15 and 20m setting respectively. This beat the uniform weight baseline by 1.5 to 2 BLEU points and the strongest single feature (LASER) baseline by 1 BLEU point. The feature based approach performed slightly better with 15 candidate runs and beat the strongest single feature baseline (LASER) by 1.3 BLEU points.

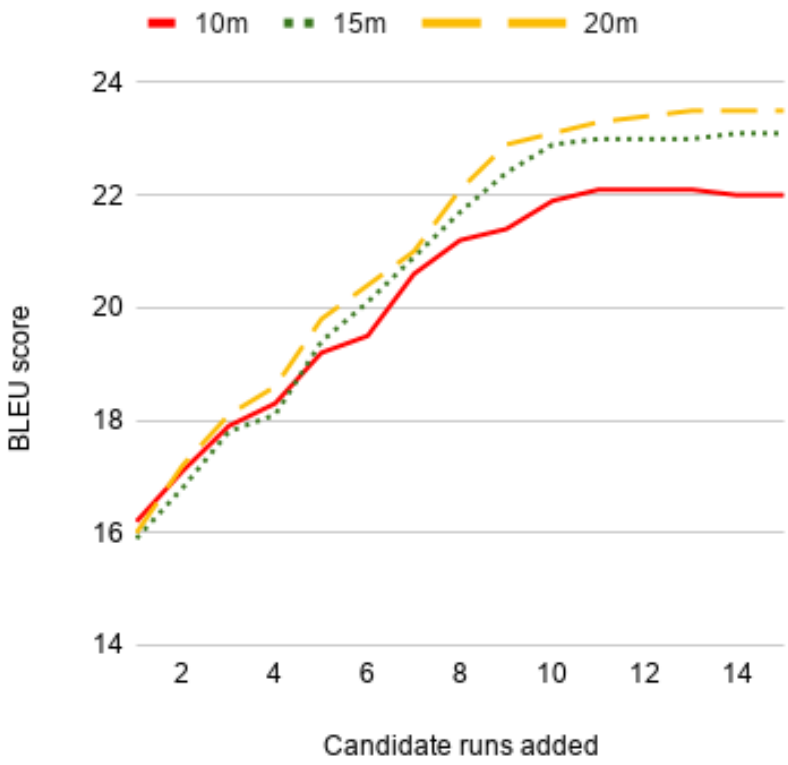


Figure 6.2: Improvement in BLEU scores of the final NMT system as data from additional 'candidate' training runs is added to the tuning stage to learn weights. Training data was filtered using the learned weights.

6.4 Analysis

The following sections examine the learned weights, the effect of transferring them to noisy corpora of a different language pair and the method's performance when exposed to specific kinds of noise.

6.4.1 Learned Weights

Table 6.3 shows the weights learned using the *tuning* network, normalized to sum to one. Unsurprisingly, the strongest feature (dual cross-entropy) has the highest weight, with the sentence length ratio and IBM Model 1 (weak multi-lingual features) drawn for the next place while source and target LM have relatively low weights.

Feature	**Weight**	**Feature**
IBM Model 1	0.07	0.12
Source LM	0.03	0.02
Target LM	0.02	0.02
Dual xent	0.81	0.76
Sen. Length Ratio	0.07	0.08

Table 6.3: Feature weights learned post-tuning with the weight-based and the feature-based approaches. The weights have been normalized to sum to 1 (column).

6.4.2 Weight Transfer

Since the feature functions we use for our experiments are reasonably language-independent, a reasonable experiment is to see if the feature weights learned on one language-pair can be transferred to a noisy corpus of another another language pair. However, we hypothesize that unless the feature distributions (proxy for noise profile of the dataset) of the datasets are similar, this transfer will have limited success.

We test this hypothesis using the Maltese-English Paracrawl corpus. The training corpus contains 26.9 million sentence pairs and was sentence aligned using Vecalign and de-duplicated in a manner similar to our primary experiments. The validation

and the test sets for these experiments are from the EUbookshop[6] dataset and contain 3k and 2.2k sentences respectively. The sentence level features were computed using the procedure described in Section 6.2.2 and we use the DGT corpus[7] (about 1.6 million parallel sentences) to the train the clean translation models, the source and the target language models.

1-Feature Filtering Baselines	
Target LM	28.3
Source LM	27.1
Dual Cross-Entropy	**32.5**
Filtering using Transfer Weights	
Uniform weight baseline	30.5
Weight based	31.6
Feature based	31.3

Table 6.4: BLEU scores for the Maltese-English Paracrawl NMT systems where the training data was filtered using single features or a *transferred* (from Estonian-English) weighted combination of features.

The results of these experiments appear in Table 6.4. Even though filtering with the transferred weights beats the simpler single feature baselines, it fails to beat the strongest one, dual cross-entropy. It is worth noting that the reason filtering with the learned weights does this well is because the dual cross-entropy feature has the highest weight from our previous experiments. These experiments suggest that some form of feature distribution matching across corpora is required before weight transfer becomes viable.

[6] `opus.nlpl.eu/EUbookshop.php`
[7] `data.europa.eu/euodp/en/data/dataset/dgt-translation-memory`

6.4.3 Sensitivity to Noise Types

Inspired by Khayrallah and Koehn (2018), we look at how the most common noisy types in the Paracrawl data set affect the performance of the proposed method. For the purpose of these experiments, we use the *Europarl* v8 [8] Estonian-English data set. The training data set consists of about 651k parallel sentences, 11.2m source and 15.7m target tokens. We only use the feature-based method for this analysis and each experiment tunes weights based on 5 *candidate* runs.

We add synthetic noise to this data set by replacing 50% of the sentences in the data set to contain a specific kind of noise. The noise types we looked at and their perturbation methods are described below:

1. Misaligned sentences: Since parallel corpora extraction efforts use automated document and sentence alignment methods, noise includes source sentences which are not aligned to the correct target sentence. To emulate this, we randomly shuffle the source sentences of half the sentences in the clean data set.

2. Misordered words: A result of automatic or imperfect human translation, we add this noise to the clean data set by randomly shuffling the words within the source sentences.

3. Wrong language: This is a very common noise type in web-crawled corpora. We

[8] `www.statmt.org/europarl`

Noise Type	% Retained
Misaligned sentences	92
Misordered words	81
Wrong language	89
Untranslated words	78

Table 6.5: The portion of the clean sentences retained after perturbing 50% of the data set with specific noise types, learning feature weights and resampling the top 50% samples.

emulate it by performing lexical replacements (from Estonian to French).

4. Untranslated words: This other common noise type is added to our data set by copying the source sentence to the target.

For each type of noise, we perform the following experiment: perturb 50% of the clean data with the chosen noise type, compute feature values for the sentences in the full data set, learn feature weights using the weight-based method described in section 6.1, filter out the top 50% of the data set and measure the percentage of *clean* (non-perturbed) sentences which were retained.[9] The results of this analysis appears in Table 6.5. The method performs significantly better than chance in all noise categories, but given our choice of features, it is better at filtering out misaligned sentences and sentences with tokens in the wrong language and is slightly less effective at dealing with misordered and untranslated words.

[9]We note that the performance of this analysis depends on the chosen features. As an extreme example, if we perturb the source sentences and only consider a target-side feature (such as target language model scores), we will have no way of discriminating bad noisy samples from the clean ones.

6.5 Synopsis

In this chapter, we present a method which allows the use of multiple sample features which are used to learn the sample usefulness for the task and the curriculum for NMT training jointly. We use this method to denoise and filter noisy parallel data for improving the performance of neural machine translation systems. We learn interpolation weights for sentence-level features by modeling and searching over the weight-reward space. These are used to score and filter sentences in the noisy corpora. Our experiments with Estonian-English Paracrawl show gains of over a BLEU point over the strongest single feature filtering and uniform weight baselines. Analysis also shows that this method is effective at addressing the most common noise types in web-crawled corpora.

Chapter 7

Conclusion

Curriculum learning hypothesizes that presenting training samples in a meaningful order to machine learners during training may help improve model quality and convergence speed. In this dissertation, we examined this framework for *meta-learning* in the context of neural machine translation (NMT). Starting from an empirical exploration of various hand-designed curricula using proxies for sample usefulness derived from human intuition, we moved on to *learning* these curricula. Next, we used multiple NMT models to learn curricula before finally allowing the use of learned usefulness metrics in our final chapter. Figure 7.1 (also presented in the introduction of this dissertation) is a summary of how these lines of work are connected.

We now move on to the primary research questions which were introduced at the beginning of this thesis and examine our research and results in this context. We will conclude with a look at some future work which can be derived from this research

	Sample Usefulness metric	Time Dependent?	Curriculum Type	Secondary Task
Empirical Exploration (Chap. 3)	Auxiliary model scores, linguistic features	No	Hand-Designed	Domain Adaptation
RL-Q-Learning (Chap. 4)	Noise CDS scores	Yes	Learned	Denoising
Multi-lingual bandits (Chap. 5)	Language ID	Yes	Learned from multiple runs	Multi-lingual translation
Reward modeling (Chap. 6)	Learned based on features	No	Learned from multiple runs	Filtering

Table 7.1: A summary of the work on learning curricula for NMT training.

and a few concluding thoughts.

7.1 Task dependent optimal curriculum

We consider the problem of designing an ideal curriculum for training an NMT model and how the properties of the task and the dataset affect this design. We showed in chapter 3 that several hand-designed curricula can match or beat the baselines. However, this process is extremely sensitive to hyperparameters and hence choosing the right curriculum can be a hard and expensive task. Chapter 4, , 5 and 6 show that instead of relying on a hand-designed curriculum, we can learn one from scratch jointly with the task. In the best case, these *learned* curricula beat the state-of-the-art results in translation performance and in the worst-case, they match the baselines but still beat the hand-designed or expensive search-based baselines.

7.2 Using human-driven design

Hand-designed curricula have seen considerable success in the training of neural machine translation systems. Specifically, fine-tuning (or continued training) and filtering are used in many state-of-the-art NMT systems. These have the advantage that they are simple to implement and analyze. However, as we show in Chapter 3, more popular curricula such as those based on the easy-to-hard methodology display limited success and only after extensive tuning. In most of our work, we have hence decided to use curricula based on human-intuition as our baselines and instead learn these curricula jointly with the NMT training procedure. As mentioned in the previous section, we have showed through our work that it is possible to *learn* these curricula which are comparable to or better than the hand-designed ones.

7.3 Sample usefulness

Finally, we consider the notion of sample usefulness for the task of training an NMT system. Defining the notion of an easy or hard example with respect to the training of machine learning models in general relies on proxy scores (which are generally sample features). We explored several usefulness scores, linguistic features and auxiliary model scores in Chapter 3, noise scores in Chapter 4 and language type in Chapter 5. We find that the features which are closest to the task at hand – e.g., noise scores for denoising, language id for multilingual training – are best to use when

a single feature is being used. In Chapter 6, we show that this usefulness metric can be learned with the curriculum as a weighted combination of several features, thus alleviating the need to specify in advance one single feature which will serve as our usefulness score.

7.4 Future work and final thoughts

We show through our work that curriculum learning for neural machine translation is a promising area of research. Through the course of this inquiry, we came across several avenues for future exploration which could possibly improve the scope and performance of the proposed frameworks. We list the major ones below:

- Rewards, observations and exploration: All agents that learn jointly with the NMT system require some form of feedback to generate training samples to train themselves. In our work, we have used validation set based performance as this reward. This is expensive to compute per update and replacing it with a more stable or time invariant reward (X. Wang et al., 2019) may help improve the performance of this method. Some possible ways of addressing this problem are:

    1. Sampled validation reward : Instead of measuring performance on the entire validation set, we could instead sample a representative batch from the validation set and use that to compute our reward. As a computational

trick, concatenating this batch to the training batch could give us rewards which are only dependent on the forward pass of the model for the previous update in one go, at the cost of having a smaller training batch (assuming finite GPU size) and a slightly delayed reward. The representative batch could be fixed or changed every few epochs. Both these approaches have limitations, the former could lead to the agent overfitting on the small validation batch, and the latter could cause instability in training because of the change in magnitude of the reward when the validation batch changes.

2. Using rewards generated from training : Similar to X. Wang et al. (2019) who use gradient based rewards, rewards generated from the training procedure of the NMT environment could remove the need for expensive validation set reward generation. However, this may lead to overfitting problems since the agent never has access to the validation set to calibrate its performance.

3. Asynchronous reward computation: Since most reinforcement learning models, such as Q-learning, only update their parameters every few updates and even then only sample from an experience replay which may not contain the sample for the most recent time step, an asynchronous reward evaluation which eventually provides a reward may be sufficient. This is similar to the approach we used in Chapter 4.

Change in environment (NMT network) parameters and their gradients may

serve as a useful alternative among others. The observations from the NMT system are another area where improvements could be made. The observation is meant to be a summary of the state of the NMT system (we use log-likelihoods of prototypical sentences) and additional work in representation learning and model compression may be applicable here. Some possible ways of addressing this are:

1. Use no observation: In our previous work we have tried to use variations of Q-learning and contextual multi-arm bandits which do not require an observation from the environment (practically, a static observation is provided at each time step). This has the effect of learning a stateless policy, one which tries to maximize the reward the agent receives without conditioning on the state of the environment and produces one optimal action (from a state independent distribution) as a policy. We determined in Chapter 4 that doing this resulted in loss of performance but it did result in an improvement in training speed. This may be a relevant approach for some tasks, especially ones where a curriculum for a relatively homogeneous dataset needs to be learned.

2. Use environment model parameters: Instead of relying on a hand-designed function to compress the state (parameters) of the environment, these parameters can be used directly. Since there are many more parameters in the model that can reasonably be expected to serve as an observation,

finding methods of compressing parameters or identifying the most salient ones (e.g., final layer parameters) may be relevant.

Additionally, to minimize the number of models which the agent uses to train itself and to avoid observation sparsity, alternative exploration techniques (we use linearly decaying epsilon-greedy) such as Thompson sampling (W. R. Thompson, 1933) could be explored.

- Improvements in reinforcement learning: Our agents in Chapters 4 and 5 use reinforcement learning (RL). We explore a relatively small portion of this actively evolving research area and any improvements from this space could improve the performance of these proposed models.

- Computational costs: While learned curricula, especially ones that use reinforcement learning can alleviate the need to hand-design data input techniques, they are computationally expensive to train. Some require training data from many NMT runs, each of which take several days of GPU time to train. Further exploration in effectively gathering and using this agent training data may help lower the computational cost of curriculum learning.

- Hybrid curriculum learning: We showed in Chapter 5 that several simple search-based curricula serve as good starting points for fine tuning. This opens up the interesting possibility of using curriculum learning to obtain good pre-trained models for hand-designed curricula, especially in low-resource scenarios and

where NMT training data is extremely heterogeneous.

- Transfer learning: The prospect of *transferring* learned curricula from one task to another is also an interesting one. This is especially useful in cases where a curriculum from a high-resource task can be transferred to a low-resource task. We explored this possibility in Chapter 6 in the context of multilingual learning for low resource languages. This technique can be explored on other NMT tasks and datasets.

In this dissertation, we explored curriculum learning for neural machine translation and show that it is in fact possible to learn orderings of the training data jointly with the NMT system to boost model performance. We see this as a first step in helping NMT systems manage their own ever increasing and heterogeneous training data.

Chapter 8

Appendix

8.1 Supplementary material for chapter 3

baseline	2.84				
	default	*reverse*	*boost*	*reduce*	*noshuffle*
one-best score	**7.1**	**9.2**	**14.7**	**7.8**	**7.8**
max wd freq(de)	2.0	1.7	**4.6**	1.6	**7.8**
max wd freq(en)	**7.9**	0.8	**8.1**	**5**	**10.8**
max wd freq(deen)	**4.3**	2.5	**4.2**	**2.9**	2.2
avg wd freq(de)	**6.5**	**4.1**	**5.5**	**7.5**	**8.8**
avg wd freq(en)	2.7	**6.8**	2.2	2.6	**5.8**
avg wd freq(deen)	1.6	**8.7**	**2.9**	1.7	**3.7**
sent len(de)	2.4	**3.0**	**2.9**	1.6	**4.6**
sent len(en)	**3.3**	**3.3**	2.5	2.1	**5.1**
sent len(deen)	2.0	2.2	2.0	2.0	**4.3**

Table 8.1: Decoding performance of different curriculum learning models at the 7th checkpoint with initial learning rate 0.0002.

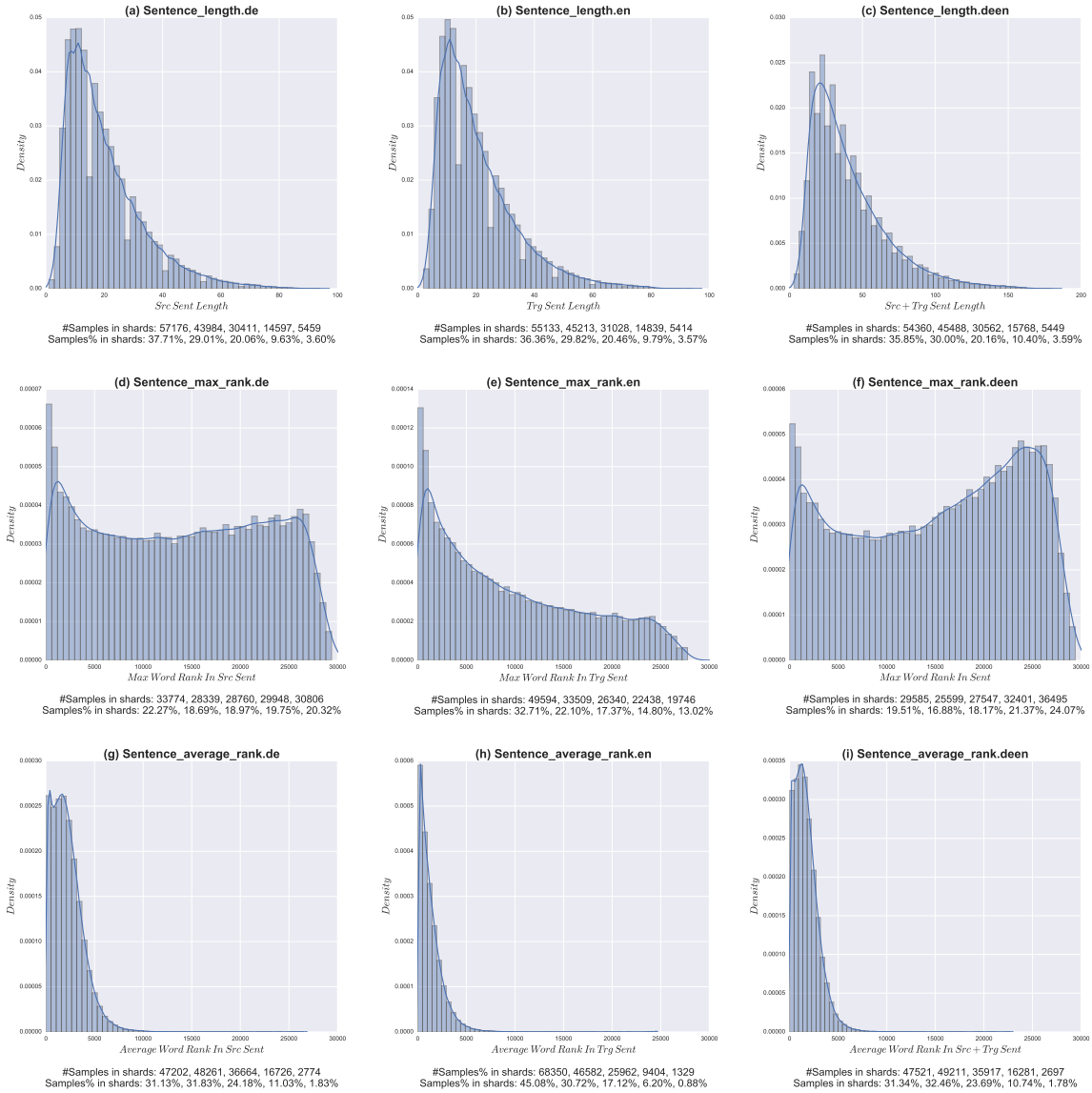


Figure 8.1: Statistics on GE-EN TED Talks training set (151,627 samples in total) scored by different difficulty criteria. We split the training data into 5 shards. Bucketing results using Jenks Natural Breaks classification algorithm are shown below each subplot, starting from easiest shard to harder shards.

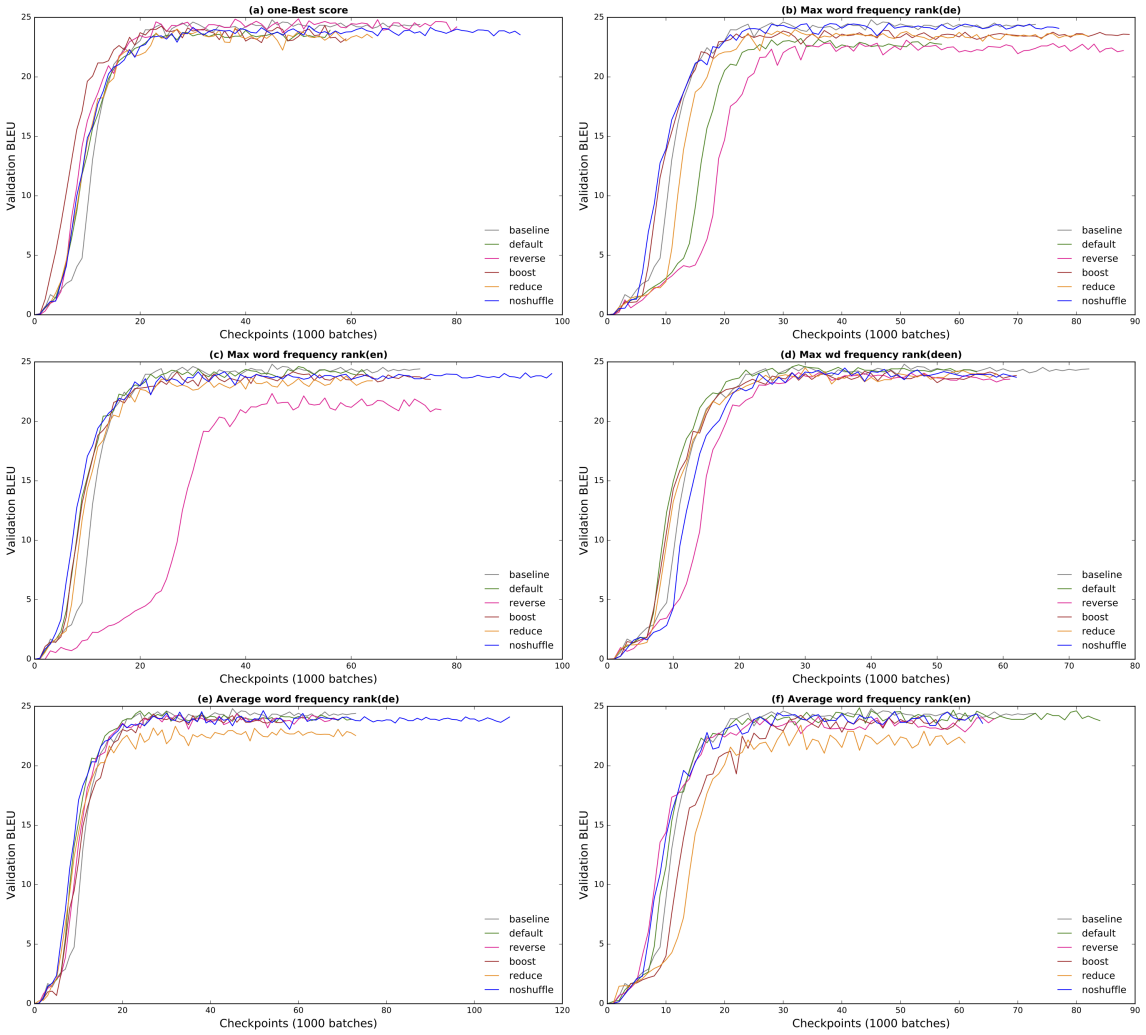


Figure 8.2: Validation BLEU curves with initial learning rate 0.0002 for different sample ranking criteria (part 1/2).

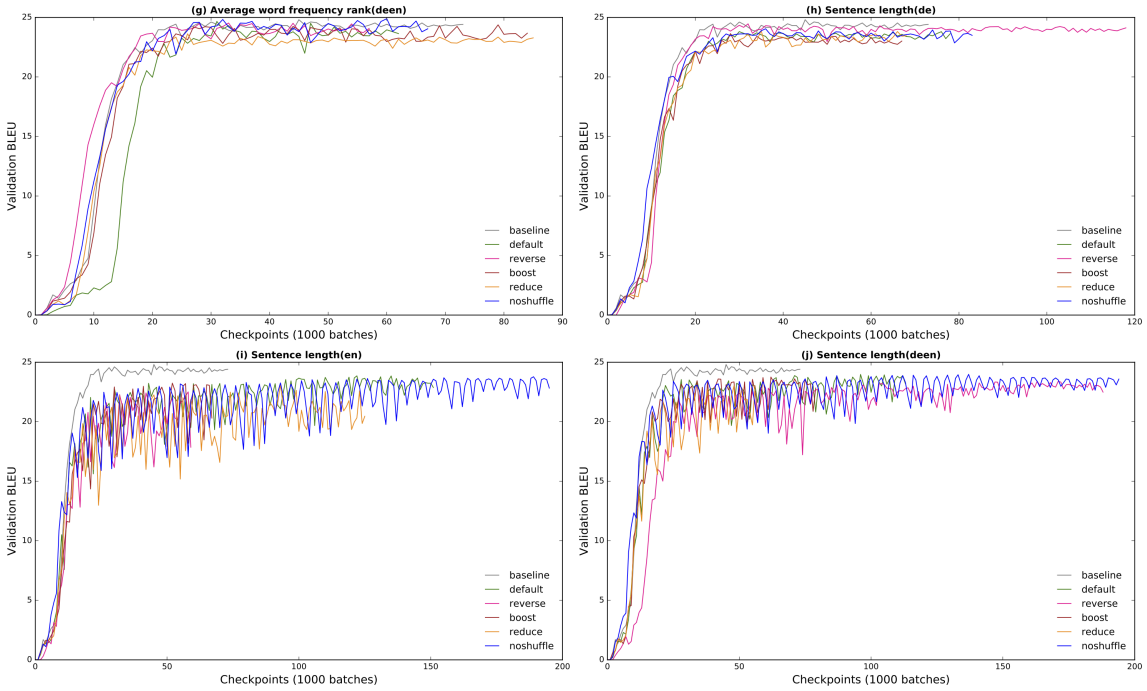


Figure 8.3: Validation BLEU curves with initial learning rate 0.0002 for different sample ranking criteria (part 2/2).

baseline	25.1				
	default	*reverse*	*boost*	*reduce*	*noshuffle*
one-best score	**28.6**	3.7	**26.5**	**26.3**	**27.8**
max wd freq(de)	18.8	0.0	**27.9**	18.6	**29.1**
max wd freq(en)	0.4	**26.7**	0.0	8.5	0.7
max wd freq(deen))	2.0	**28.3**	**28.3**	0.2	5.2
avg wd freq(de)	24.1	**28.8**	23.3	23.1	2.3
avg wd freq(en)	18.1	21.5	1.9	9.7	4.9
avg wd freq(deen)	**25.2**	**26.4**	18.5	2.0	**26.4**
sent len(de)	9.9	0.0	24.3	17.6	**30.0**
sen len(en)	**26.6**	18.6	5.3	**26.1**	24.2
sent len(deen)	1.1	14.4	24.1	**26.0**	24.2

Table 8.2: Decoding performance of different curriculum learning models at the 7th checkpoint with initial learning rate 0.0008.

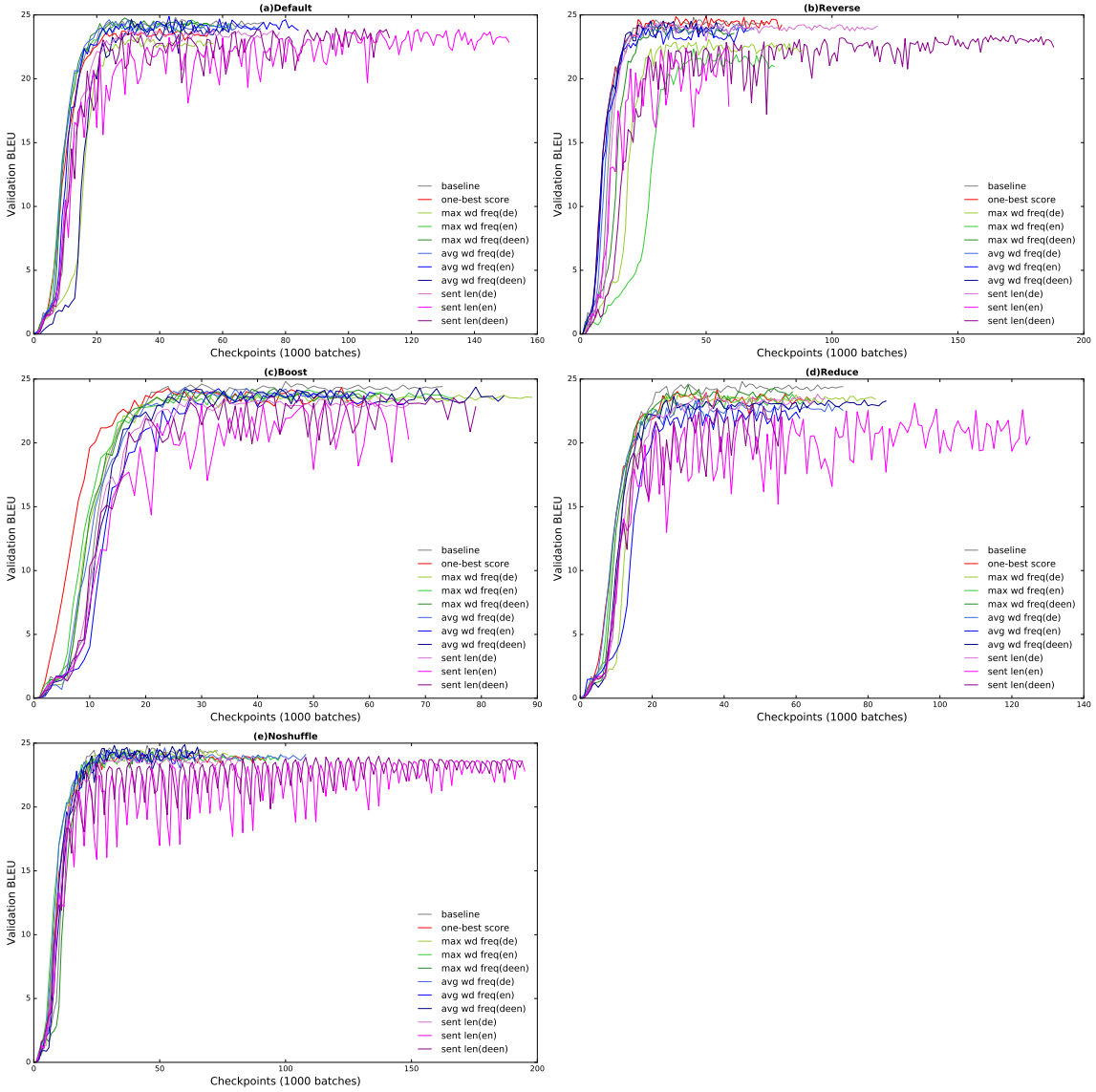


Figure 8.4: Validation BLEU curves with initial learning rate 0.0002 for different curriculum schedules.

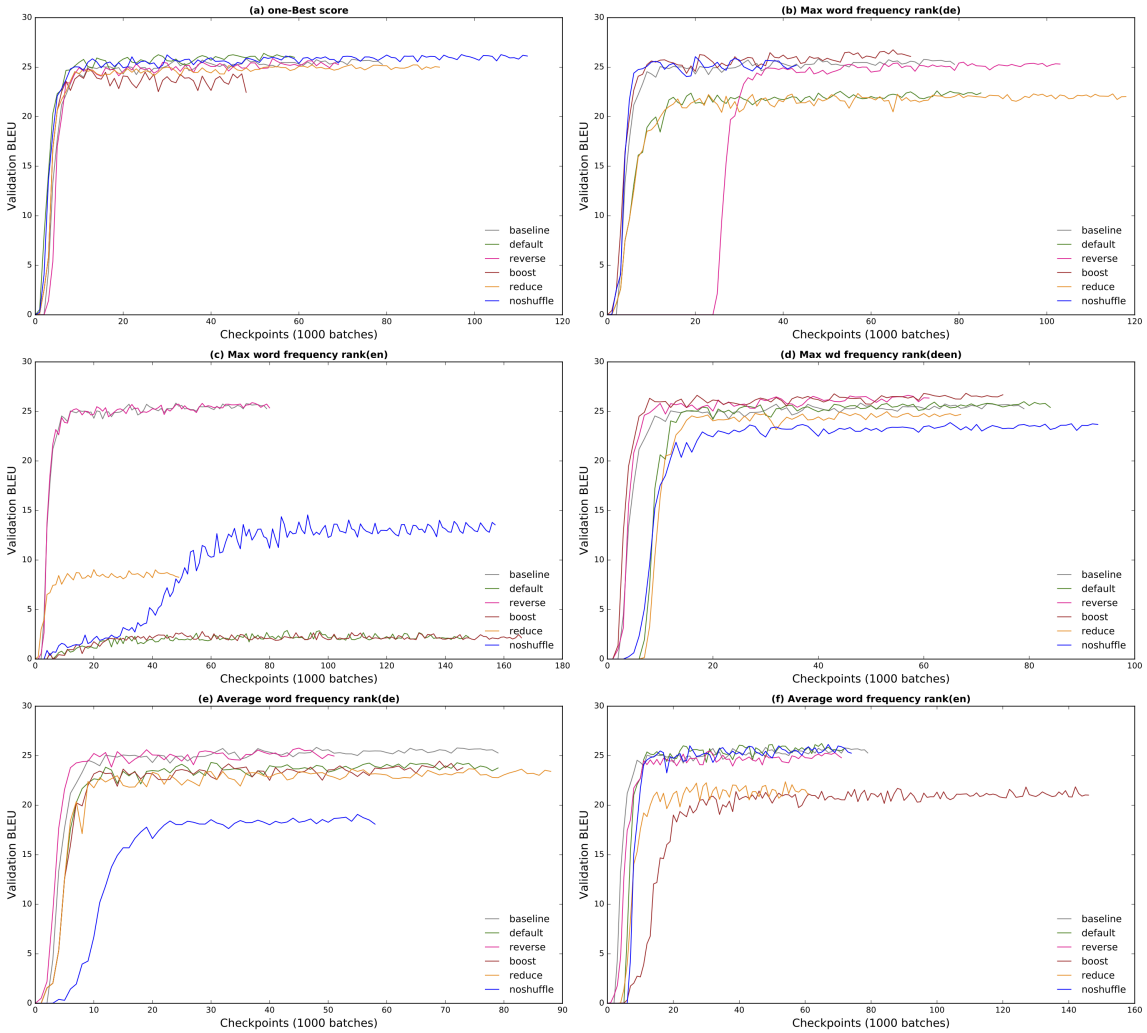


Figure 8.5: Validation BLEU curves with initial learning rate 0.0008 for different sample ranking criteria (part 1/2).

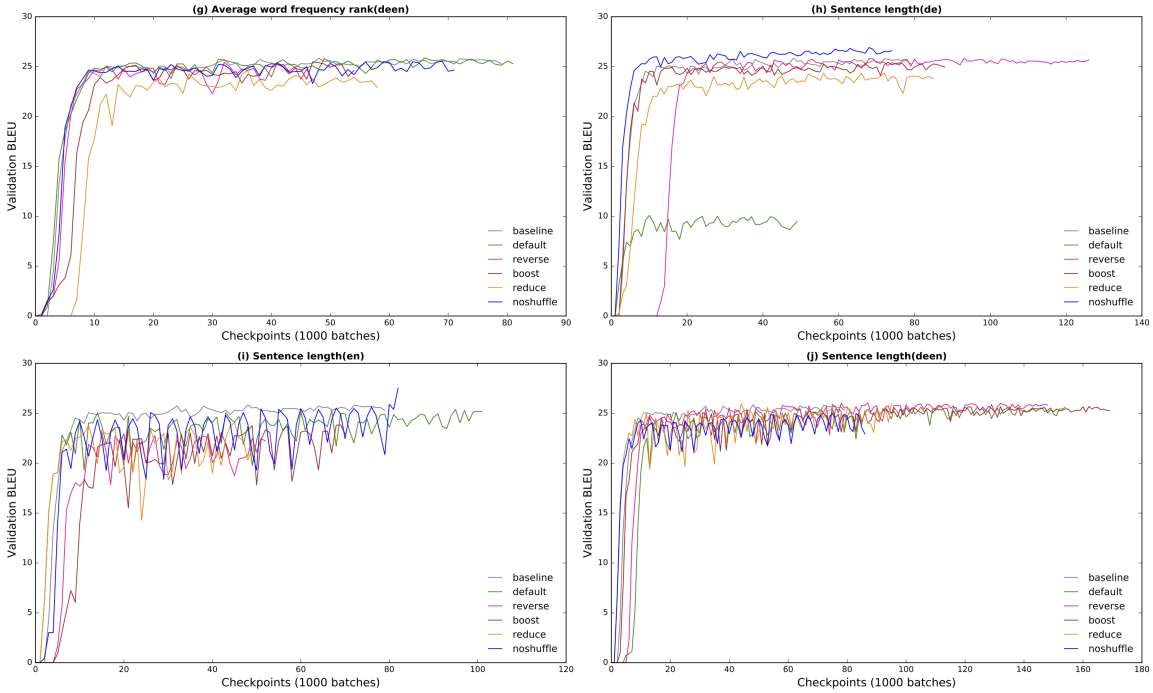


Figure 8.6: Validation BLEU curves with initial learning rate 0.0008 for different sample ranking criteria (part 2/2).

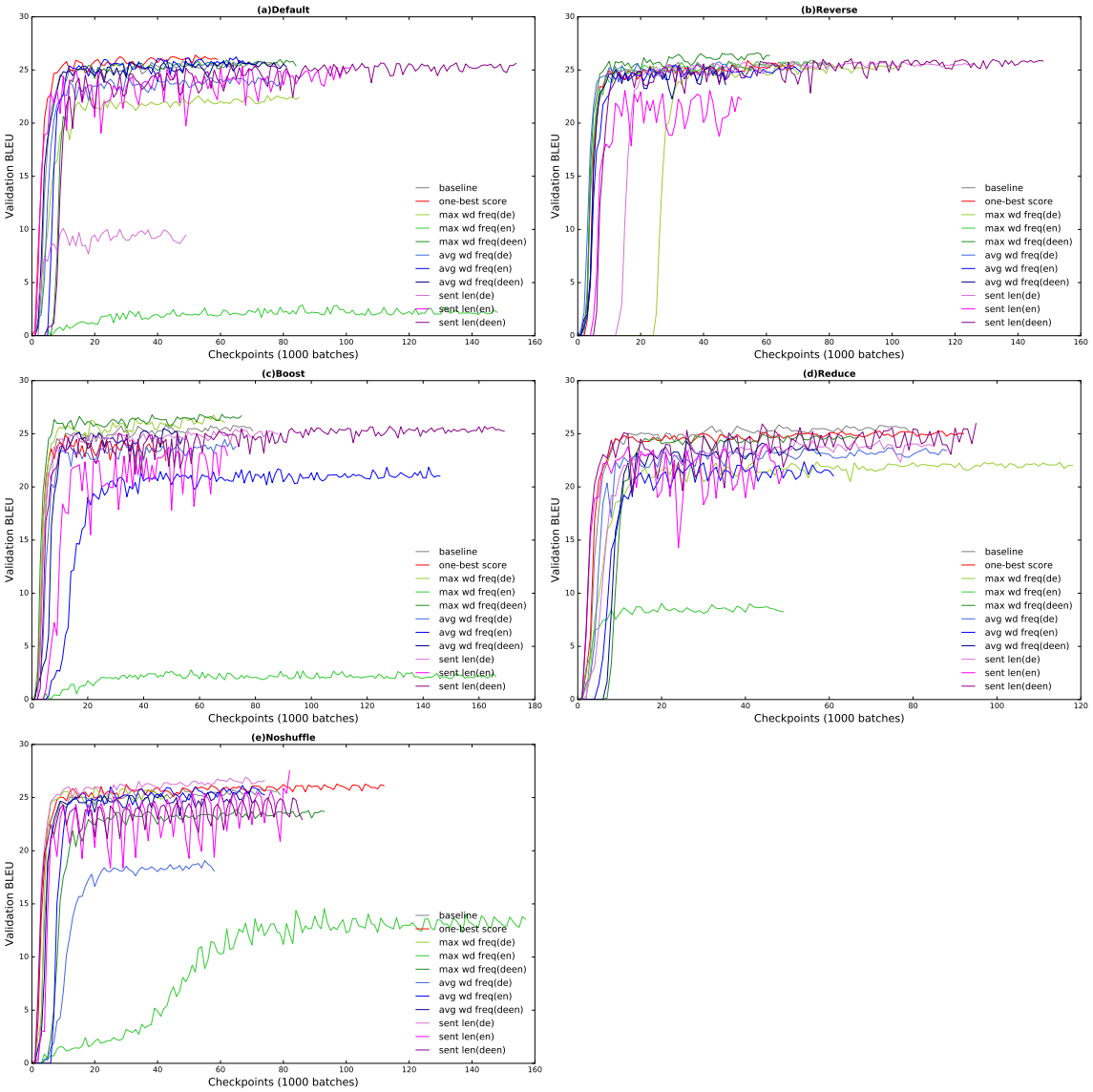


Figure 8.7: Validation BLEU curves with initial learning rate 0.0008 for different curriculum schedules.

8.2 Supplementary material for chapter 4

8.2.1 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
- 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.

8.3 Supplementary material for chapter 5

8.3.1 Contextual MAB hyper-parameters

- Observations: We sample 32 prototype sentences from each bin to create a *prototype batch* of 192 sentences.
- MABs: We use 5 or 10 MABs which were MLPs with 2 x 256-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.
- Minimum number of transitions in replay buffer before training starts: 3000

- The window for the delta-perplexity reward was 1.

Bibliography

Aharoni, Roee, Melvin Johnson, and Orhan Firat (June 2019). "Massively Multilingual 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 and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 3874–3884. URL: https://www.aclweb.org/anthology/N19-1388.

Amiri, Hadi, Timothy Miller, and Guergana Savova (2017). "Repeat before Forgetting: Spaced Repetition for Efficient and Effective Training of Neural Networks". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2401–2410.

Antonova, Alexandra and Alexey Misyurev (June 2011). "Building a Web-Based Parallel Corpus and Filtering Out Machine-Translated Text". In: *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*. Portland, Oregon: Association for Computational Linguistics, pp. 136–144. URL: https://www.aclweb.org/anthology/W11-1218.

BIBLIOGRAPHY

Arivazhagan, Naveen, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George F. Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu (2019). "Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges". In: *CoRR* abs/1907.05019. arXiv: 1907.05019. URL: http://arxiv.org/abs/1907.05019.

Ash, T. (1989). "Dynamic node creation in backpropagation networks". In: *Connection Science*, pp. 365–375.

Auer, Peter, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire (Jan. 2003). "The Nonstochastic Multiarmed Bandit Problem". In: *SIAM J. Comput.* 32.1, pp. 48–77. URL: https://doi.org/10.1137/S0097539701398375.

Axelrod, Amittai, Xiaodong He, and Jianfeng Gao (2011a). "Domain Adaptation via Pseudo In-Domain Data Selection". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 355–362. URL: http://www.aclweb.org/anthology/D11-1033.

Axelrod, Amittai, Xiaodong He, and Jianfeng Gao (2011b). "Domain Adaptation via Pseudo In-domain Data Selection". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Association for Computational Linguistics, pp. 355–362. URL: http://dl.acm.org/citation.cfm?id=2145432.2145474.

BIBLIOGRAPHY

Baker, C.L. (1979). "Syntactic theory and the projection problem". In: *Linguistic Inquiry, 10*, pp. 533–581.

Bañón, Marta, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L. Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, Sergio Ortiz Rojas, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Elsa Sarrías, Marek Strelec, Brian Thompson, William Waites, Dion Wiggins, and Jaume Zaragoza (July 2020). "ParaCrawl: Web-Scale Acquisition of Parallel Corpora". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 4555–4567. URL: https://www.aclweb.org/anthology/2020.acl-main.417.

Bapna, Ankur and Orhan Firat (Nov. 2019). "Simple, Scalable Adaptation for Neural Machine Translation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 1538–1548. URL: https://www.aclweb.org/anthology/D19-1165.

Bellman, Richard (1957). *Dynamic Programming*. Dover Publications.

Bengio, Yoshua (June 2012). "Practical Recommendations for Gradient-Based Training of Deep Architectures". In: *arXiv:1206.5533 [cs]*. arXiv: 1206.5533 [cs].

BIBLIOGRAPHY

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert, and Jason Weston (2009). "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: ACM, pp. 41–48.

Bernier-Colborne, Gabriel and Chi-kiu Lo (Aug. 2019). "NRC Parallel Corpus Filtering System for WMT 2019". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy: Association for Computational Linguistics, pp. 252–260. URL: https://www.aclweb.org/anthology/W19-5434.

Bowerman, M. (1987). "The "no negative evidence" problem: How do children avoid constructing an overly general grammar?" In: *J.A. Hawkins (Ed.), Explaining language universals. Oxford: Blackwell.*

Brewer, Cynthia A (2006). "Basic mapping principles for visualizing cancer data using geographic information systems (GIS)". In: *American journal of preventive medicine* 30.2, S25–S36.

Britz, Denny, Quoc Le, and Reid Pryzant (2017). "Effective domain mixing for neural machine translation". In: *Proceedings of the Second Conference on Machine Translation*, pp. 118–126.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer (1993). "The Mathematics of Statistical Machine Translation: Parameter Estimation". In: *Computational Linguistics* 19.2, pp. 263–311. URL: https://www.aclweb.org/anthology/J93-2003.

BIBLIOGRAPHY

Chaudhary, Vishrav, Yuqing Tang, Francisco Guzmán, Holger Schwenk, and Philipp Koehn (Aug. 2019). "Low-Resource Corpus Filtering Using Multilingual Sentence Embeddings". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy: Association for Computational Linguistics, pp. 261–266. URL: https://www.aclweb.org/anthology/W19-5435.

Chen, Boxing, 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*, pp. 40–46.

Chen, Mia Xu, 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)*. Melbourne, Australia: Association for Computational Linguistics, pp. 76–86. URL: http://aclweb.org/anthology/P18-1008.

Chih-Chun Wang, S. R. Kulkarni, and H. V. Poor (2005). "Bandit problems with side observations". In: *IEEE Transactions on Automatic Control* 50.3, pp. 338–355.

Chrysochoou, Maria, Kweku Brown, Geeta Dahal, Catalina Granda-Carvajal, Kathleen Segerson, Norman Garrick, and Amvrossios Bagtzoglou (2012). "A GIS and

indexing scheme to screen brownfields for area-wide redevelopment planning". In: *Landscape and Urban Planning* 105.3, pp. 187–198.

Cirik, Volkan, Eduard Hovy, and Louis-Philippe Morency (Nov. 2016). "Visualizing and Understanding Curriculum Learning for Long Short-Term Memory Networks". In: *arXiv:1611.06204 [cs]*. eprint: 1611.06204 (cs).

Collier, Mark and Hector Urdiales Llorens (2018). "Deep Contextual Multi-armed Bandits". In: *CoRR* abs/1807.09809. arXiv: 1807.09809. URL: http://arxiv.org/abs/1807.09809.

Collobert, Ronan and Jason Weston (2008). "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, pp. 160–167. URL: http://doi.acm.org/10.1145/1390156.1390177.

Dabre, Raj, Atsushi Fujita, and Chenhui Chu (Nov. 2019). "Exploiting Multilingualism through Multistage Fine-Tuning for Low-Resource Neural Machine Translation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 1410–1416. URL: https://www.aclweb.org/anthology/D19-1146.

BIBLIOGRAPHY

Dakwale, Praveen and Christof Monz (2017). "Fine-tuning for neural machine translation with limited degradation across in-and out-of-domain data". In: *Proceedings of the 16th Machine Translation Summit (MT-Summit 2017)*.

Doetsch, Patrick, Pavel Golik, and Hermann Ney (2017). "A comprehensive study of batch construction strategies for recurrent neural networks in MXNet". In: *arXiv preprint arXiv:1705.02414*.

Duh, Kevin (2018). *The Multitarget TED Talks Task*. `http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/`.

Duh, Kevin, 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)*. Sofia, Bulgaria, pp. 678–683. URL: `http://www.phontron.com/paper/duh13acl.pdf`.

Durrani, Nadir, Hassan Sajjad, Shafiq R. Joty, and Ahmed Abdelali (2016). "A Deep Fusion Model for Domain Adaptation in Phrase-based MT". In: *COLING*. ACL, pp. 3177–3187.

Elman, Jeffrey L. (1993). "Learning and development in neural networks: the importance of starting small". In: *Cognition* 48.1, pp. 71–99. URL: `http://www.sciencedirect.com/science/article/pii/0010027793900584`.

ElNokrashy, Muhammad N., Amr Hendy, Mohamed Abdelghaffar, Mohamed Afify, Ahmed Tawfik, and Hany Hassan Awadalla (2020). *Score Combination for Im-*

proved Parallel Corpus Filtering for Low Resource Conditions. arXiv: 2011.07933 [cs.CL].

Erdmann, Grant and Jeremy Gwinnup (Aug. 2019). "Quality and Coverage: The AFRL Submission to the WMT19 Parallel Corpus Filtering for Low-Resource Conditions Task". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy: Association for Computational Linguistics, pp. 267–270. URL: https://www.aclweb.org/anthology/W19-5436.

Esplà -Gomis, Miquel, VÃctor M. SÃ¡nchez-Cartagena, Jaume Zaragoza-Bernabeu, and Felipe SÃ¡nchez-MartÃnez (Nov. 2020). "Bicleaner at WMT 2020: Universitat d'Alacant-Prompsit's submission to the parallel corpus filtering shared task". In: *Proceedings of the Fifth Conference on Machine Translation*. Online: Association for Computational Linguistics, pp. 950–956. URL: https://www.aclweb.org/anthology/2020.wmt-1.107.

Fahlman, S.E. and C. Lebiere (1990). "he cascade-correlation learning architecture". In: *D.S. Touretzky (Ed.), Advances in neural information processing Systems*. San Mateo, CA: Morgan Kaufmann, pp. 524–532.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *CoRR* abs/1703.03400. arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

BIBLIOGRAPHY

Foster, George, Cyril Goutte, and Roland Kuhn (2010). "Discriminative Instance Weighting for Domain Adaptation in Statistical Machine Translation". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. EMNLP '10. Cambridge, Massachusetts: Association for Computational Linguistics, pp. 451–459. URL: http://dl.acm.org/citation.cfm?id=1870658.1870702.

Freitag, Markus and Yaser Al-Onaizan (2016). "Fast Domain Adaptation for Neural Machine Translation". In: *CoRR* abs/1612.06897. arXiv: 1612.06897. URL: http://arxiv.org/abs/1612.06897.

Gold, E.M. (1967). "Language identification in the limit". In: *Information and Control, 16*, pp. 447–474.

González-Rubio, Jesús (Aug. 2019). "Webinterpret Submission to the WMT2019 Shared Task on Parallel Corpus Filtering". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy: Association for Computational Linguistics, pp. 271–276. URL: https://www.aclweb.org/anthology/W19-5437.

Graves, Alex, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu (2017a). "Automated Curriculum Learning for Neural Networks". In: *International Conference on Machine Learning*, pp. 1311–1320.

Graves, Alex, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu (2017b). "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, pp. 1311–1320. URL: http://proceedings.mlr.press/v70/graves17a.html.

Gu, Jiatao, Hany Hassan, Jacob Devlin, and Victor OK Li (2018). "Universal Neural Machine Translation for Extremely Low Resource Languages". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Vol. 1, pp. 344–354.

Guzmán, Francisco, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato (Nov. 2019). "The FLORES Evaluation Datasets for Low-Resource Machine Translation: Nepali–English and Sinhala–English". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 6098–6111. URL: https://www.aclweb.org/anthology/D19-1632.

Hangya, Viktor and Alexander Fraser (Oct. 2018). "An Unsupervised System for Parallel Corpus Filtering". In: *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. Belgium, Brussels: Association for Computational Linguistics, pp. 882–887. URL: https://www.aclweb.org/anthology/W18-6477.

BIBLIOGRAPHY

Hasler, Eva, Adrià de Gispert, Felix Stahlberg, Aurelien Waite, and Bill Byrne (2017). "Source sentence simplification for statistical machine translation". In: *Computer Speech & Language* 45, pp. 221–235.

Hieber, Felix, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post (2017). "Sockeye: A toolkit for neural machine translation". In: *arXiv preprint arXiv:1712.05690*.

Jean, Sébastien, Orhan Firat, and Melvin Johnson (2018). "Adaptive Scheduling for Multi-Task Learning". In: *Continual learning Workshop, Thirty-second Conference on Neural Information Processing Systems*.

Jenks, George F. (1997). *Optimal Data Classification for Choropleth Maps. Occasional Paper No.2*. Dept. Geography, Univ. Kansas, p. 24.

Johnson, Melvin, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. (2017a). "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation". In: *Transactions of the Association of Computational Linguistics* 5.1, pp. 339–351.

Johnson, Melvin, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viegas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2017b). "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation". In: *Transactions of the As-*

sociation for Computational Linguistics 5, pp. 339–351. URL: https://www.aclweb.org/anthology/Q17-1024.

Junczys-Dowmunt, Marcin (Oct. 2018). "Dual Conditional Cross-Entropy Filtering of Noisy Parallel Corpora". In: *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. Belgium, Brussels: Association for Computational Linguistics, pp. 888–895. URL: https://www.aclweb.org/anthology/W18-6478.

Khayrallah, Huda and Philipp Koehn (July 2018). "On the Impact of Various Types of Noise on Neural Machine Translation". In: *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. Melbourne, Australia: Association for Computational Linguistics, pp. 74–83. URL: https://www.aclweb.org/anthology/W18-2709.

Khomenko, Viacheslav, Oleg Shyshkov, Olga Radyvonenko, and Kostiantyn Bokhan (2016). "Accelerating Recurrent Neural Network Training Using Sequence Bucketing and Multi-GPU Data Parallelization". In: *IEEE First International Conference on Data Stream Mining & Processing (DSMP)*. IEEE, pp. 100–103.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

Kiperwasser, Eliyahu and Miguel Ballesteros (2018a). "Scheduled Multi-Task Learning: From Syntax to Translation". In: *arXiv preprint arXiv:1804.08915*.

Kiperwasser, Eliyahu and Miguel Ballesteros (Apr. 2018b). "Scheduled Multi-Task Learning: From Syntax to Translation". en. In: *Transactions of the Association for Computational Linguistics* 6.0, pp. 225–240.

Kobus, Catherine, Josep Crego, and Jean Senellart (2016). "Domain control for neural machine translation". In: *arXiv preprint arXiv:1612.06140*.

Kocmi, Tom and Ondrej Bojar (2017). "Curriculum Learning and Minibatch Bucketing in Neural Machine Translation". In: *Recent Advances in Natural Language Processing (RANLP)*. eprint: 1707.09533.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst (2007). "Moses: Open Source Toolkit for Statistical Machine Translation". In: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. ACL '07. Prague, Czech Republic: Association for Computational Linguistics, pp. 177–180. URL: http://dl.acm.org/citation.cfm?id=1557769.1557821.

Kreutzer, Julia and Stefan Riezler (2019). *Self-Regulated Interactive Sequence-to-Sequence Learning*. arXiv: 1907.05190 [cs.CL].

Krueger, Kai A and Peter Dayan (2009). "Flexible shaping: how learning in small steps helps". In: *Cognition* 110.3, pp. 380–394. URL: https://doi.org/10.1016/j.cognition.2008.11.014.

BIBLIOGRAPHY

Kumar, Gaurav, George Foster, Colin Cherry, and Maxim Krikun (June 2019). "Reinforcement Learning based Curriculum Optimization for 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 and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 2054–2061. URL: https://www.aclweb.org/anthology/N19-1208.

Kumar, Gaurav, Philipp Koehn, and Sanjeev Khudanpur (2021a). *Learning Feature Weights using Reward Modeling for Denoising Parallel Corpora*. arXiv: 2103.06968 [cs.CL].

Kumar, Gaurav, Philipp Koehn, and Sanjeev Khudanpur (2021b). *Learning Policies for Multilingual Training of Neural Machine Translation Systems*. arXiv: 2103.06964 [cs.CL].

Kumar, M., Benjamin Packer, and Daphne Koller (2010). "Self-Paced Learning for Latent Variable Models". In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2010/file/e57c6b956a6521b28495f2886ca0977a-Paper.pdf.

Kurfalı, Murathan and Robert Östling (Aug. 2019). "Noisy Parallel Corpus Filtering through Projected Word Embeddings". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy:

Association for Computational Linguistics, pp. 277–281. URL: https://www.aclweb.org/anthology/W19-5438.

Lakew, Surafel Melaku, Quintino F. Lotito, Matteo Negri, Marco Turchi, and Marcello Federico (2018). "Improving Zero-Shot Translation of Low-Resource Languages". In: *CoRR* abs/1811.01389. arXiv: 1811.01389. URL: http://arxiv.org/abs/1811.01389.

Langford, John and Tong Zhang (2008). "The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt, D. Koller, Y. Singer, and S. Roweis. Vol. 20. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2007/file/4b04a686b0ad13dce35fa99fa4161c65-Paper.pdf.

Liu, Ming, 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*, pp. 334–344. URL: https://aclanthology.info/papers/K18-1033/k18-1033.

Liu, Xuebo, Houtim Lai, Derek F. Wong, and Lidia S. Chao (2020). "Norm-Based Curriculum Learning for Neural Machine Translation". In: *CoRR* abs/2006.02014. arXiv: 2006.02014. URL: https://arxiv.org/abs/2006.02014.

BIBLIOGRAPHY

Luong, Minh-Thang and Christopher D. Manning (2015). "Stanford Neural Machine Translation Systems for Spoken Language Domain". In: *International Workshop on Spoken Language Translation*. Da Nang, Vietnam.

Matsoukas, Spyros, Antti-Veikko I. Rosti, and Bing Zhang (2009). "Discriminative Corpus Weight Estimation for Machine Translation". In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*. EMNLP '09. Singapore: Association for Computational Linguistics, pp. 708–717. URL: http://dl.acm.org/citation.cfm?id=1699571.1699605.

Mnih, Volodymyr, 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 (Feb. 2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533. URL: http://dx.doi.org/10.1038/nature14236.

Moore, Robert C. and William Lewis (2010). "Intelligent Selection of Language Model Training Data". In: *Proceedings of the ACL 2010 Conference Short Papers*. ACLShort '10. Association for Computational Linguistics, pp. 220–224. URL: http://dl.acm.org/citation.cfm?id=1858842.1858883.

Newport, E.L. (1988). "Constraints on learning and their role in language acquisition: Studies of the acquisition of American Sign Language". In: *Language Sciences*, pp. 147–172.

Newport, E.L. (1990). "Maturational constraints on language learning." In: *Cognitive Science*, pp. 11–28.

Ng, Nathan, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov (Aug. 2019). "Facebook FAIR's WMT19 News Translation Task Submission". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Florence, Italy: Association for Computational Linguistics, pp. 314–319. URL: https://www.aclweb.org/anthology/W19-5333.

Ott, Myle, 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*.

Pandey, S., D. Agarwal, D. Chakrabarti, and V. Josifovski (2007). "Bandits for taxonomies: a model-based approach". In: *SIAM Data Mining Conference*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318. URL: http://www.aclweb.org/anthology/P02-1040.

Parcheta, Zuzanna, Germán Sanchis-Trilles, and Francisco Casacuberta (Aug. 2019). "Filtering of Noisy Parallel Corpora Based on Hypothesis Generation". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared*

Task Papers, Day 2). Florence, Italy: Association for Computational Linguistics, pp. 282–288. URL: https://www.aclweb.org/anthology/W19-5439.

Peris, Álvaro and Francisco Casacuberta (2018). "Active Learning for Interactive Neural Machine Translation of Data Streams". In: *CoRR* abs/1807.11243. arXiv: 1807.11243. URL: http://arxiv.org/abs/1807.11243.

Peterson, G. B. (2004). "A day of great illumination: B. F. Skinner's discovery of shaping." In: pp. 317–328.

Pinker, S. (1989). *Learnability and cognition*. Cambridge, MA: MIT Press.

Plunkett, K. and V. Marchman (1990). *From rote learning to system building*. Center for Research in Language, TR 9020, University of California, San Diego.

Rarrick, Spencer, Chris Quirk, and Will Lewis (Sept. 2011). "MT Detection in Web-Scraped Parallel Corpora". In: *Proceedings of MT Summit XIII*. Asia-Pacific Association for Machine Translation. URL: https://www.microsoft.com/en-us/research/publication/mt-detection-in-web-scraped-parallel-corpora/.

Richards, Jack C. (1984). "Language Curriculum Development". In: *RELC Journal* 15.1, pp. 1–29. eprint: https://doi.org/10.1177/003368828401500101. URL: https://doi.org/10.1177/003368828401500101.

Rohde, D. L. and D. C. Plaut (1994). "Language acquisition in the absence of explicit negative evidence: how important is starting small?" In: *Cognition*. Vol. 72. 1, pp. 67–109.

Rossenbach, Nick, Jan Rosendahl, Yunsu Kim, Miguel Graça, Aman Gokrani, and Hermann Ney (Oct. 2018). "The RWTH Aachen University Filtering System for the WMT 2018 Parallel Corpus Filtering Task". In: *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. Belgium, Brussels: Association for Computational Linguistics, pp. 946–954. URL: https://www.aclweb.org/anthology/W18-6487.

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

Sánchez-Cartagena, Víctor M., Marta Bañón, Sergio Ortiz-Rojas, and Gema Ramírez-Sánchez (Oct. 2018). "Prompsit's submission to WMT 2018 Parallel Corpus Filtering shared task". In: *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*. Brussels, Belgium: Association for Computational Linguistics.

Sanger, T. D. (1994). "Neural network learning control of robot manipulators using gradually increasing task difficulty". In: *IEEE Trans. on Robotics and Automation*.

Schapire, Robert E. (2002). "The Boosting Approach to Machine Learning: An Overview". In: *MSRI Workshop on Nonlinear Estimation and Classification*.

Schwenk, Holger and Matthijs Douze (Aug. 2017). "Learning Joint Multilingual Sentence Representations with Neural Machine Translation". In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Associ-

ation for Computational Linguistics, pp. 157–167. URL: https://www.aclweb.org/anthology/W17-2619.

Sen, Sukanta, Asif Ekbal, and Pushpak Bhattacharyya (Aug. 2019). "Parallel Corpus Filtering Based on Fuzzy String Matching". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy: Association for Computational Linguistics, pp. 289–293. URL: https://www.aclweb.org/anthology/W19-5440.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde (2017). "Nematus: A Toolkit for Neural Machine Translation". In: *Proceedings of the Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Sennrich, Rico, 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)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. URL: http://www.aclweb.org/anthology/P16-1162.

Shah, Kashif, Loïc Barrault, and Holger Schwenk (2010). "Translation Model Adaptation by Resampling". In: *WMT@ACL*. Association for Computational Linguistics, pp. 392–399.

Shultz, T.R. and W.C. Schmidt (1991). "A cascade-correlation model of balance scale phenomenon". In: *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Skinner, B. F. (1958). "Reinforcement today". In: *American Psychologist, 13*, pp. 94–99.

Soares, Felipe and Marta R. Costa-jussà (Aug. 2019). "Unsupervised corpus filtering and mining". In: *Proceedings of the Fourth Conference on Machine Translation*. Florence, Italy: Association for Computational Linguistics.

Spitkovsky, Valentin I., Hiyan Alshawi, and Daniel Jurafsky (2010). "From Baby Steps to Leapfrog: How "Less Is More" in Unsupervised Dependency Parsing". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 751–759.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Thompson, Brian and Philipp Koehn (Nov. 2019). "Vecalign: Improved Sentence Alignment in Linear Time and Space". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 1342–1348. URL: https://www.aclweb.org/anthology/D19-1136.

Thompson, William R. (Dec. 1933). "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". In: *j-BIOMETRIKA* 25.3/4, pp. 285–294. URL: http://www.jstor.org/stable/2332286.

Tokic, Michel and Günther Palm (2011). "Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax". In: *Proceedings of the 34th Annual German Conference on Advances in Artificial Intelligence*. KI'11. Berlin, Germany: Springer-Verlag, pp. 335–346.

Tsvetkov, Yulia, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer (2016). "Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning". In:

Venugopal, Ashish, Jakob Uszkoreit, David Talbot, Franz Och, and Juri Ganitkevitch (July 2011). "Watermarking the Outputs of Structured Prediction with an application in Statistical Machine Translation." In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 1363–1372. URL: https://www.aclweb.org/anthology/D11-1126.

Wan, Yu, Baosong Yang, Derek F. Wong, Yikai Zhou, Lidia S. Chao, Haibo Zhang, and Boxing Chen (2020). "Self-Paced Learning for Neural Machine Translation". In: *CoRR* abs/2010.04505. arXiv: 2010.04505. URL: https://arxiv.org/abs/2010.04505.

Wang, Rui, 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*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1482–1488. URL: http://aclweb.org/anthology/D17-1155.

Wang, Rui, Masao Utiyama, and Eiichiro Sumita (2018). "Dynamic Sentence Sampling for Efficient Training of Neural Machine Translation". en. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2, pp. 298–304.

Wang, Wei, Isaac Caswell, and Ciprian Chelba (2019). "Dynamically Composing Domain-Data Selection with Clean-Data Selection by "Co-Curricular Learning" for Neural Machine Translation". In: *CoRR* abs/1906.01130. arXiv: 1906.01130. URL: http://arxiv.org/abs/1906.01130.

Wang, Wei, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba (2018a). "Denoising Neural Machine Translation Training with Trusted Data and Online Data Selection". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, pp. 133–143. URL: http://aclweb.org/anthology/W18-6314.

Wang, Wei, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba (Oct. 2018b). "Denoising Neural Machine Translation Training with Trusted Data and Online Data Selection". In: *Proceedings of the Third Conference on Machine*

Translation: Research Papers. Brussels, Belgium: Association for Computational Linguistics, pp. 133–143. URL: https://www.aclweb.org/anthology/W18-6314.

Wang, Xinyi, Hieu Pham, Paul Michel, Antonios Anastasopoulos, Graham Neubig, and Jaime G. Carbonell (2019). "Optimizing Data Usage via Differentiable Rewards". In: *CoRR* abs/1911.10088. arXiv: 1911.10088. URL: http://arxiv.org/abs/1911.10088.

Watkins, Christopher J. C. H. and Peter Dayan (1992). "Q-learning". In: *Machine Learning* 8.3, pp. 279–292. URL: https://doi.org/10.1007/BF00992698.

Wees, Marlies van der, Arianna Bisazza, and Christof Monz (2017a). "Dynamic Data Selection for Neural Machine Translation". en. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1400–1410.

Wees, Marlies van der, Arianna Bisazza, and Christof Monz (2017b). "Dynamic Data Selection for Neural Machine Translation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1400–1410. URL: http://aclweb.org/anthology/D17-1147.

Wexler, K. and P. Cullicover (1980). *Forma/principles of language acquisition*. Cambridge, MA: MIT Press.

Wu, Jiawei, 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), pp. 1252–1262. URL: https://aclanthology.info/papers/N18-1113/n18-1113.

Xu, Hainan and Philipp Koehn (Sept. 2017). "Zipporah: a Fast and Scalable Data Cleaning System for Noisy Web-Crawled Parallel Corpora". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 2945–2950. URL: https://www.aclweb.org/anthology/D17-1319.

Yeo and R.A. Johnson (2000). *A New Family of Power Transformations to Improve Normality or Symmetry*.

Zhang, Dakun, Jungi Kim, Josep Crego, and Jean Senellart (2017). "Boosting Neural Machine Translation". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 271–276. URL: http://www.aclweb.org/anthology/I17-2046.

Zhang, Xuan, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J. Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat (2018). "An Empirical Exploration of Curriculum Learning for Neural Machine Translation". In: *CoRR* abs/1811.00739. arXiv: 1811.00739. URL: http://arxiv.org/abs/1811.00739.

BIBLIOGRAPHY

Zhang, Xuan, 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.

Zhou, Yikai, Baosong Yang, Derek F. Wong, Yu Wan, and Lidia S. Chao (July 2020). "Uncertainty-Aware Curriculum Learning for Neural Machine Translation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 6934–6944. URL: https://www.aclweb.org/anthology/2020.acl-main.620.

Zoph, Barret, Deniz Yuret, Jonathan May, and Kevin Knight (Nov. 2016). "Transfer Learning for Low-Resource Neural Machine Translation". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1568–1575. URL: https://www.aclweb.org/anthology/D16-1163.

Vita

Gaurav Kumar

noisychannel.github.io

gkumar@cs.jhu.edu

Interests

Machine Translation, Automatic Speech Recognition, Natural Language Processing and Machine Learning.

Education

The Johns Hopkins University 2014-2021

Research Assistant (Center for Language and Speech Processing)

Ph.D. in Computer Science

Research Advisors : Dr. Sanjeev Khudanpur, Dr. Philipp Koehn

VITA

The Johns Hopkins University 2012-2014

M.S.E. in Computer Science

Research Advisors : Dr. Sanjeev Khudanpur, Dr. Jason Eisner

Vellore Institute of Technology University, India 2004-2008

B.Tech. in Computer Science and Engineering

Publications

1. Gaurav Kumar, Philipp Koehn, and Sanjeev Khudanpur, **Learning Feature Weights using Reward Modeling for Denoising Parallel Corpora** in *arXiv preprint*: 2103.06968, 2021 [PDF]

2. Gaurav Kumar, Philipp Koehn, and Sanjeev Khudanpur, **Learning Policies for Multilingual Training of Neural Machine Translation Systems** in *arXiv preprint*: 2103.06964, 2021 [PDF]

3. Gaurav Kumar, George Foster, Colin Cherry, Maxim Krikun, **Reinforcement Learning based Curriculum Optimization for Neural Machine Translation** in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics* (NAACL), 2019 [PDF]

4. Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, Kevin Duh, **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* (NAACL), 2019 [PDF]

5. Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, Marine Carpuat, **An Empirical Exploration of Curriculum Learning for Neural Machine Translation** in *arXiv preprint*: 1811.00739, 2018 [PDF]

6. Huda Khayrallah, Gaurav Kumar, Kevin Duh, Matt Post, Philipp Koehn, **Neural Lattice Search for Domain Adaptation in Machine Translation** in *Proceedings of the Eighth International Joint Conference on Natural Language Processing* (IJCNLP), 2017 [PDF]

7. Jan Trmal, Gaurav Kumar, Vimal Manohar, Sanjeev Khudanpur, Matt Post, Paul McNamee, **Using of heterogeneous corpora for training of an ASR system**, *arXiv preprint*: 1706.00321, 2017 [PDF]

8. Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, Pengcheng Yin, **DyNet: The Dynamic Neural Net-**

work Toolkit, *arXiv preprint*:1701.03980, 2017 [PDF]

9. Shuoyang Ding, Huda Khayrallah, Philipp Koehn, Matt Post, Gaurav Kumar, and Kevin Duh, **The JHU Machine Translation Systems for WMT 2017** in *Proceedings of the Second Conference on Machine Translation* (WMT), 2017 [PDF]

10. Gaurav Kumar, Graeme Blackwood, Jan Trmal, Daniel Povey and Sanjeev Khudanpur, **"A Coarse-Grained Model for Optimal Coupling of ASR and SMT Systems for Speech Translation"** in *Conference on Empirical Methods in Natural Language Processing* (EMNLP), 2015 [PDF]

11. Matt Post and Yuan Cao and Gaurav Kumar, **"Joshua 6: A phrase-based and hierarchical statistical machine translation system"** in *Prague Bulletin of Mathematical Linguistics*, 2015 [PDF]

12. Gaurav Kumar, Yuan Cao, Ryan Cotterell, Chris Callison-Burch, Daniel Povey and Sanjeev Khudanpur, **"Translations of the CALLHOME Egyptian Arabic Corpus for Conversational Speech Translation"** in *Proceedings of the International Workshop on Spoken Language Translation* (IWSLT), 2014 [PDF]

13. Gaurav Kumar, Matt Post, Daniel Povey and Sanjeev Khudanpur, **"Some Insights from Translating Conversational Telephone Speech"** in *IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014 [PDF]

14. Sarana Nutanong, Yanif Ahmad, I-Jeng Wang, Jeliazko Jeliazkov, Gaurav Kumar and Thomas B. Woolf, **Learning about transitions: Adaptive Controls for the Molecular Dynamics Database** in *The 58th Annual Meeting of the Biophysical Society*, 2014

15. Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch and Sanjeev Khudanpur, **"Improved Speech-to-Text Translation with the Fisher and Callhome Spanish-English Speech Translation Corpus"** in *Proceedings of the International Workshop on Spoken Language Translation* (IWSLT), 2013 [PDF]

Experience

Research Assistant, **CLSP, Johns Hopkins University**, Baltimore 2013-2021

Worked on problems related to Machine Translation and Speech Recognition.

Software Developer Intern, **Google AI (Translate)**, Montreal 2018-2018

Developed a Reinforcement-learning based technique for noisy data selection for Neural Machine Translation.

VITA

Summer Research Intern, **IBM T.J. Watson Research Center**, NY 2014-2014

Worked on research involving optimal couplings for Machine Translation and Automatic Speech Recognition systems.

Lead Architect, Team Lead, **Blisstering Solutions Pvt. Ltd.**, India 2008-2012

Led a team which developed multi-media recommendation engines and web-application frameworks for a variety of clients.

Product Manager, Team Lead, **Banyan Mobile Pvt. Ltd.**, India 2010-2012

Built recommendation engines and web-application frameworks for telecommunication companies.

CTO, Trustee, **Immunize India Charitable Trust** 2010-

Responsible for the technology platform which supports the vaccination efforts.

Other Employment / Workshops

Jelinek Summer Workshop on Speech and Language Technology 2017

Neural Machine Translation with Minimum Parallel Resources

Supervisors: Dr. George Foster, Dr. Colin Cherry

Shared Task, Conference on Machine Translation (WMT) 2017

VITA

Neural Lattice Search Methods

Third Machine Translation Marathon in the Americas (MTMA) 2017

Low and zero resource Neural Machine Translation

Jelinek Summer Workshop on Speech and Language Technology 2015

Context-aware Neural Machine Translation

Supervisor: Dr. Chris Dyer

Summer Camp for Applied Language Exploration (SCALE) 2015

Speech Translation Methods for Low Resource Languages

Supervisors: Dr. Matt Post, Dr. Sanjeev Khudanpur

First Machine Translation Marathon in the Americas (MTMA) 2015

Explicit Context Encoding with Recurrent Architectures

DARPA Broad Operational Language Translations (BOLT) 2014

Speech Translation for Egyptian-Arabic

Summer Camp for Applied Language Exploration (SCALE) 2013

Improved Speech Translation Architectures

Supervisors: Dr. Matt Post, Dr. Sanjeev Khudanpur

Relevant Open Source Contributions

Kaldi Speech Recognition Toolkit

Joshua Statistical Machine Translation Toolkit

Dynet Neural Network Toolkit

VITA

Skills

Programming Languages: Python, Java, Perl, Bash, PHP, C, C++, LaTeX

Deep Learning Frameworks: PyTorch, Tensorflow, MXnet

Deep Learning toolkits: Fairseq, Lingvo, Sockeye, OpenNMT, Theano, Kaldi

Updated on May 20, 2021.