The UMD Neural Machine Translation Systems at WMT17 Bandit Learning Task

Amr Sharaf and **Shi Feng** and **Khanh Nguyen** and **Kianté Brantley** and **Hal Daumé III**
Department of Computer Science
University of Maryland, College Park
{amr,shifeng,kxnguyen,kdbrant,hal}@cs.umd.edu

Abstract

We describe the University of Maryland machine translation systems submitted to the WMT17 German-English Bandit Learning Task. The task is to adapt a translation system to a new domain, using only *bandit feedback*: the system receives a German sentence to translate, produces an English sentence, and only gets a scalar score as feedback. Targeting these two challenges (adaptation and bandit learning), we built a standard neural machine translation system and extended it in two ways: (1) robust reinforcement learning techniques to learn effectively from the bandit feedback, and (2) domain adaptation using data selection from a large corpus of parallel data.

1 Introduction

We describe the University of Maryland systems for bandit machine translation. For the shared translation task of the EMNLP 2017's second conference on machine translation (WMT17), we focused on the task of bandit machine translation. This shared task was set up, consistent with (Kreutzer et al., 2017), simultaneously as a bandit learning problem *and* a domain adaptation problem. This raises the natural question: can we combine these potentially complementary information sources?

To investigate this question, we started from a standard neural machine translation (NMT) setup §2[1], and then we:

1. applied domain adaptation techniques by data selection (Moore and Lewis, 2010) to the out-of-domain data, with the goals of filtering out harmful data and fine-tuning the training process to focus only on relevant sentences (§4).
2. trained robust reinforcement learning algorithms that can effectively learn from bandit feedback (§3); this allows our model to "test" proposed generalizations and adapt from the provided feedback signals.

Tackling the problem of learning with bandit feedback is important because neural machine translation systems, like other natural language processing technology, currently learn almost exclusively from labeled data for a specific domain. While this approach is useful, it cannot scale to a broad variety of language and domains, as linguistic systems often cannot generalize well beyond their training data. Machine translation systems need to be able to learn to improve their performance from naturalistic interaction with users in addition to labeled data.

Bandit feedback (Robbins, 1985) offers systems the opportunity to "test" proposed generalizations and receive feedback on their performance; particularly interesting are *contextual* bandit systems, which make predictions based on a given input context (Auer et al., 2002; Langford and Zhang, 2008; Beygelzimer et al., 2010; Dudik et al., 2011). For example, a neural translation system trained on parliament proceedings often performs quite poorly at translating anything else. However, a translation system that is deployed to facilitate conversations between users might receive either explicit feedback (e.g. thumbs up/down) on its translations, or even implicit feedback, for example, the conversation partner asking for clarifications. There has recently been a flurry of work specifically addressing the bandit structured prediction problem (Chang et al., 2015; Sokolov et al., 2016a,b), of which machine translation is a special case.

[1] Our implementation is based on OpenNMT (Klein et al., 2017), an open-source toolkit for neural MT.

Because this task is—at it's core—a domain adaptation problem (for which a bandit learning signal is available to "help"), we also explored the use of standard domain adaptation techniques. We make a strong assumption that a sizable amount of *monolingual, source language* data is available *before* bandit feedback begins.[2] We believe that in many realistic settings, one can at least get some amount of unlabeled data to begin with (we consider $40k$ sentences). Using this monolingual data, we use data selection on a large corpus of parallel out-of-domain data (Europarl, NewsCommentary, CommonCrawl, Rapid) to seed an initial translation model.

Overall, the results support the following conclusions (§5), based on the limited setting of one new domain and one language pair:

1. data selection for domain adaptation alone improves translation quality by about 1.5 BLEU points.

2. on *top* of the domain adaptation, reinforcement learning (which requires exploration) leads to an *initial* degradation of about 3 BLEU points, which is recovered (on development data) after approximately $40k$ sentences of bandit feedback.[3]

One limitation of our current setup is that we used bandit feedback on development data to train a "critic" function for our reinforcement learning implementation, which, in the worst case, means that our results over-estimate performance on the first $120k$ examples (more details in §5.3).

2 Neural MT architecture

We closely follow Luong et al. (2015) for the structure of our neural machine translation (NMT) systems. Our NMT model consists of an encoder and a decoder, each of which is a recurrent neural network (RNN). We use a bi-directionaral RNN as the encoder and a uni-directional RNN as the decoder. The model directly estimates the posterior distribution $P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$ of translating a source sentence $\boldsymbol{x} = (x_1, \cdots, x_n)$ to a target sentence $\boldsymbol{y} = (y_1, \cdots, y_m)$:

$$P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{m} P_\theta(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \quad (1)$$

where $\boldsymbol{y}_{<t}$ are all tokens in the target sentence prior to y_t.

Each local distribution $P_\theta(y \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$ is modeled as a multinomial distribution over the target language vocabulary. We represent this as a linear transformation followed by a softmax function on the decoder's output vector $\tilde{\boldsymbol{h}}_t^{dec}$:

$$P_\theta(y \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s\, \tilde{\boldsymbol{h}}_t^{dec}; \tau) \quad (2)$$

$$\tilde{\boldsymbol{h}}_t^{dec} = \tanh(\boldsymbol{W}_o[\boldsymbol{h}_t^{dec}; \boldsymbol{c}_t]) \quad (3)$$

$$\boldsymbol{c}_t = \text{attend}(\boldsymbol{h}_{1:n}^{enc}, \boldsymbol{h}_t^{dec}) \quad (4)$$

where $[.;.]$ is the concatenation of two vectors, $\text{attend}(.,.)$ is an attention mechanism, [4], τ is the temperature hyperparameter of the softmax function, $\boldsymbol{h}^{enc}$ and $\boldsymbol{h}^{dec}$ are the hidden vectors generated by the encoder and the decoder, respectively.

During training, the encoder first encodes $\boldsymbol{x}$ to a continuous vector $\boldsymbol{\Phi}(\boldsymbol{x})$, which is used as the initial hidden vector for the decoder. The decoder performs RNN updates to produce a sequence of hidden vectors:

$$\begin{aligned} \boldsymbol{h}_0^{dec} &= \boldsymbol{\Phi}(\boldsymbol{x}) \\ \boldsymbol{h}_t^{dec} &= f_\theta\left(\boldsymbol{h}_{t-1}^{dec}, \left[\tilde{\boldsymbol{h}}_{t-1}^{dec}; \boldsymbol{e}(y_t)\right]\right) \end{aligned} \quad (5)$$

where $\boldsymbol{e}(.)$ is a word embedding lookup operation, f_θ is an LSTM cell. [5]

At prediction time, the ground-truth token y_t in Eq. 5 is replaced by the model's own prediction $\hat{y}_t$:

$$\hat{y}_t = \arg\max_y P_\theta(y \mid \hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) \quad (6)$$

In a supervised learning framework, an NMT model is typically trained under the maximum log-likelihood objective:

$$\mathcal{L}_{sup}(\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_{\text{tr}}}\left[\log P_\theta\left(y \mid \boldsymbol{x}\right)\right] \quad (7)$$

where D_{tr} is the training set.

However, this learning framework is not applicable to our problem since reference translations are not available.

[2] This raises a natural question: in the cases where this assumption is unreasonable, could we do adaptation online?

[3] Unfortunately, due to our implementation bug, our evaluation of the test server is incomplete for the reinforcement learning setting; see §5.3 for a discussion.

[4] We use the "concat" mechanism in (Luong et al., 2015).

[5] Feeding $\tilde{\boldsymbol{h}}_t^{dec}$ to the next step is "input feeding."

3 Reinforcement Learning

The translation process of an NMT model can be viewed as a Markov decision process operating on a continuous state space. The states are the hidden vectors $\boldsymbol{h}_t^{dec}$ generated by the decoder. The action space is the target language's vocabulary.

3.1 Markov decision process formulation

To generate a translation from a source sentence $\boldsymbol{x}$, an NMT model commences at an initial state $\boldsymbol{h}_0^{dec}$, which is a representation of $\boldsymbol{x}$ computed by the encoder. At time step $t > 0$, the model decides the next action to take by defining a stochastic policy $P_\theta(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$, which is directly parametrized by the parameters θ of the model. This policy takes the previous state $\boldsymbol{h}_{t-1}^{dec}$ as input and produces a probability distribution over all actions (words in the target vocabulary). The next action $\hat{y}_t$ is chosen either by taking $\arg\max$ or sampling from this policy. The encoder computes the current state $\boldsymbol{h}_t^{dec}$ by applying an RNN update on the previous state $\boldsymbol{h}_{t-1}^{dec}$ and the next action taken $\hat{y}_t$ (Eq. 5).

The objective of bandit NMT is to find a policy that maximizes the expected quality of translations sampled from the model's policy:

$$\mathcal{L}_{pg}(\theta) = \mathbb{E}_{\substack{\boldsymbol{x}\sim D_{\text{tr}} \\ \hat{\boldsymbol{y}}\sim P_\theta(\boldsymbol{y}|\boldsymbol{x})}}\Big[R(\hat{\boldsymbol{y}}, \boldsymbol{x})\Big] \tag{8}$$

where R is a reward function that returns a score in $[0, 1]$ reflecting the quality of the input translation.

We optimize this objective function by policy gradient methods. The gradient of the objective in Eq. 8 with respect to θ is: [6]

$$\begin{aligned}\nabla_\theta \mathcal{L}_{pg}(\theta) &= \mathbb{E}_{\hat{\boldsymbol{y}}\sim P(\cdot)}\left[R(\hat{\boldsymbol{y}})\nabla_\theta \log P_\theta(\hat{\boldsymbol{y}})\right] \\ &= \sum_{t=1}^{m} \mathbb{E}_{\substack{\hat{y}_t\sim \\ P(\cdot|\hat{\boldsymbol{y}}_{<t})}}\Big[R(\hat{\boldsymbol{y}})\nabla_\theta \log P_\theta(\hat{y}_t \mid \hat{\boldsymbol{y}}_{<t})\Big]\end{aligned} \tag{9}$$

3.2 Advantage Actor-Critic

Algorithm 1 The A2C algorithm for NMT.

1: **for** $k = 0 \cdots K$ **do**
2: receive a source sentence $\boldsymbol{x}$
3: sample a translation: $\hat{\boldsymbol{y}} \sim P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$
4: receive reward $R(\hat{\boldsymbol{y}}, \boldsymbol{x})$
5: update the NMT model using the gradient in Eq. 9
6: update the critic model using the gradient in Eq. 12
7: **end for**

We follow the approach of the advantage actor-critic (A2C) algorithm (Mnih et al., 2016), which combines the REINFORCE algorithm (Williams, 1992) with actor-critic. The algorithm approximates the gradient in Eq. 9 by a single-point sample and normalize the rewards by V values to reduce variance:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{pg}(\theta) &\approx \sum_{t=1}^{m} \nabla_\theta \log P_\theta(\hat{y}_t \mid \hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) \bar{R}_t(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) \\ &\text{with } \bar{R}_t(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) \equiv R(\hat{\boldsymbol{y}}, \boldsymbol{x}) - V(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x})\end{aligned} \tag{10}$$

where $\hat{y}_t \sim P(\cdot \mid \hat{\boldsymbol{y}}_{<t}, \boldsymbol{x})$ and $V(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) = \mathbb{E}\left[R(\hat{\boldsymbol{y}}, \boldsymbol{x}) \mid \hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}\right]$ is a baseline that estimates the expected future reward given $\boldsymbol{x}$ and $\hat{\boldsymbol{y}}_{<t}$.

We train a critic model V_ω to estimate the V values. This model is an attention-based encoder-decoder model that encodes a source sentence $\boldsymbol{x}$ and decodes a predicted translation $\hat{\boldsymbol{y}}$. At time step t, it computes $V_\omega(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) = \boldsymbol{W}_o\, \tilde{\boldsymbol{h}}_t^{dec}$ where $\tilde{\boldsymbol{h}}_t^{dec}$ is the hidden state of the RNN decoder, and $\boldsymbol{W}_o$ is a matrix that transforms a vector into a scalar. [7]

The critic model is trained to minimize the MSE between its estimates and the true values:

$$\mathcal{L}_{crt}(\omega) = \mathbb{E}_{\boldsymbol{x}\sim D_{\text{tr}}}\left[\sum_{t=1}^{m} \left\|R(\hat{\boldsymbol{y}}, \boldsymbol{x}) - V_\omega(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x})\right\|^2\right] \tag{11}$$

Given a fixed $\boldsymbol{x}$, the gradient with respect to ω of this objective is:

$$\nabla_\omega \mathcal{L}_{crt}(\omega) = \sum_{t=1}^{m} \left[R(\hat{\boldsymbol{y}}) - V_\omega(\hat{\boldsymbol{y}}_{<t})\right] \nabla_\omega V_\omega(\hat{\boldsymbol{y}}_{<t}) \tag{12}$$

Algorithm 1 describes our algorithm. For each $\boldsymbol{x}$, we draw a single sample $\hat{\boldsymbol{y}}$ from the NMT model, which is used for both estimating the gradient of the NMT model (Eq. 10) and the gradient of the critic model (Eq. 12). We update the NMT model and the critic model simultaneously.

4 Domain Adaptation

We performed domain adaptation by choosing the best out-of-domain parallel data for training using Moore and Lewis (2010) cross-entropy based data selection technique.

Cross-Entropy Difference

The Moore and Lewis method uses the cross-entropy difference $H_I(s)$ - $H_O(s)$ for scoring a

[6] For notation brevity, we omit $\boldsymbol{x}$ from this equation. The expectations are also taken over all given $\boldsymbol{x}$.

[7] We abuse the notation $\tilde{\boldsymbol{h}}^{dec}$ to denote the decoder output. But since the translation model and the critic model do not share parameters, their decoder outputs are distinct.

given sentence s, based on an in-domain language model LM_I and an out-of-domain language model LM_O (Moore and Lewis, 2010). We trained LM_O using the German-English Europarl, NewsCommentary, CommonCrawl and Rapid (i.e. out-of-domain) data sets and LM_I using the e-commerce domain data provided by Amazon. After training both language models, we follow Moore and Lewis method by applying the cross-entropy difference to score each sentence in the out-of-domain data. The cross-entropy is mathematically defined as:

$$H(W) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{LM}(w_i|w_1, \cdots, w_{i-1})$$

where P_{LM} is the probability of a LM for the word sequence W and $w_1, \cdots, w_{i-1}$ represents the history of the word w_i.

Sentences with the lowest cross-entropy difference scores are the most relevant because they are the more similar to the in-domain data and less similar to the average of the out-of-domain data. Using this criteria, the top n out-of-domain sentences are used to create the training set D_{tr}. In this work we consider various n sizes, selecting the n that provides the best performance on the validation set.

5 Experiments

This section describes the experiments we conducted in attempt to assess the challenges posed by bandit machine translation and our exploration of efficient algorithms to improve machine translation systems using bandit feedback.

As explained in previous sections, this task requires performing domain adaptation for machine translation through bandit feedback. With this in mind, we experimented with two types of models: simple domain adaptation without using the feedbacks, and reinforcement learning models that leverage the feedbacks. In the following sections, we explain how we train the regular NMT model, how we select training data for domain adaptation, and how we use reinforcement learning to leverage the bandit feedbacks.

We trained our systems using the out-of-domain parallel data restricted by the shared task. The entire out-of-domain dataset contains 4.5 millions parallel German-English sentences from Europarl, NewsCommentary, CommonCrawl and Rapid data for the News Translation (constrained) task. Our NMT model is based on OpenNMT's (Klein et al., 2017) PyTorch implementation of attention-based encoder-decoder model. We extended their implementation and added our implementation of the A2C algorithm. Details of the model configuration and training hyperparameters are listed in Table 1.

Word embedding size	500
Hidden vector size	500
Number of LSTM layers	2
Batch size	64
Epochs	13
Optimizer	SGD
Initial learning rate	1
Dropout	0.3
BPE size	20000
Vocab size	∼25k (*)

Table 1: NMT model's training hyperparameters. (*) with BPE we no longer need to prune the vocabulary, and the exact size depends on the training data.

5.1 Subword Unit for Neural Machine Translation

Neural machine translation (NMT) relies on first mapping each word into the vector space, and traditionally we have a word vector corresponding to each word in a fixed vocabulary. Due to the data scarcity, it's hard for the system to learn high quality representations for rare words. To address this problem, with the goal of open vocabulary NMT, Sennrich et al. (2015) proposed to learn subword units and perform translation on a subword level. We incorporated this approach in our system as a preprocessing step. We generate the so-called byte-pair encoding (BPE), which is a mapping from words to subword units, on the whole training set (WMT15), for both the source and target languages. The same mapping is used for all the training sets in our system. After the translation, we do an extra post-processing step to convert the target language subword units back to words. With BPE, the vocabulary size is reduced dramatically and we no longer need to prune the vocabularies. We find this approach to be very helpful and use it for all our systems.

5.2 Domain Adaptation

As explained in Section 4, we use the data selection method of (Moore and Lewis, 2010) for domain adaptation. We use the kenlm toolkit (Heafield, 2011) to build all the language models used for the data selection. We train 4-gram language models. For computing the cross-entropy similarity scores, we use the XenC (Rousseau, 2013) open source data selection tool. We use the mono-lingual data selection mode of XenC on the in-domain and out-of-domain source sentences.

We have two parameters in this data selection process: the size of in-domain dataset that is used for training the in-domain language model, and the size of the out-of-domain training data that we select. We experimented with different configurations and the results on the development server are listed in Table 2. For obtaining the in-domain data, we pre-fetch the source sentences from development and training servers. For the training server, we do not have enough keys to test all combinations, so we picked several configurations and for each sentence, we select randomly a system to translate it. In addition, we also compare with and without beam search. The purpose for this is to provide another comparable baseline for the later reinforcement learning model, for which beam search cannot be used. Thus, the domain adaptation system that we submit to the training server is the uniformly random combination of 6 systems, and their individual average BLEU scores are listed in Table 3.

It can be seen from these results that most configurations of data selection improve the overall BLEU score. The model without data selection achieves 18.70 BLEU on the development server, while the best data selection configurations achieves 20.16, while on the training server the scores are 18.65 without data selection and 20.13 with. It can also be seen from Table 3 that beam search does help with improving the BLEU score.

5.3 Reinforcement Learning Results

While translating with the domain adaptation models to the development server, we collect 320,000 triples of (source sentence, translation, feedback) from 8 submitted systems. We use these triples to pre-train the critic in the A2C algorithm. We use the same pre-trained critic for all A2C-trained systems. The critic for each model is then

	in-domain size		
o.o.d.%	**40k**	**200k**	**800k**
10%	18.50	18.57	18.85
20%	19.56	19.41	19.23
30%	19.54	**20.16**	19.11
40%	**19.58**	19.37	19.36
60%	18.88	18.81	**19.59**
85%	19.12	18.69	18.26
(*) 100%	18.70	18.70	18.70

Table 2: average BLEU scores of domain adaptation systems on the development server with different combinations of in-domain size (x-axis) and the percentage of out-of-domain data selected (y-axis). (*) we show the BLEU score of using all the out-of-domain data, do data selection performed for this row.

i.d. size	**o.o.d. %**	**beam=1**	**beam=5**
0	100%	18.07	18.65 (+0.58)
40k	40%	18.77	19.51 (+0.74)
200k	30%	**19.67**	**20.13 (+0.46)**

Table 3: Average BLEU scores of domain adaptation systems on the training server with different combinations of in-domain size, out-of-domain percentage, beam size, and the corresponding BLEU scores.

updated jointly with the actor respectively. We use Adam (Kingma and Ba, 2014) with learning rate of 10^{-4} to update the both the translation model and the critic model. We do not use dropout (Srivastava et al., 2014) during training with A2C as it makes learning less stable.

We note that there are some drawbacks when using the A2C algorithm when it comes to generating translations. Normally we generate translations by greedy decoding, which means at each time step we pick the word with the highest probability from the distribution produced by the model. But with A2C, we need to sample from the distribution of words to ensure exploration. As a direct consequence, it is not clear how to apply beam search for A2C (and for policy gradient methods in general). To control the trade-off between exploration and exploitation, we use the temperature hyperparameter τ in the softmax function. In our experiments τ is set to $\frac{2}{3}$, which produces a more

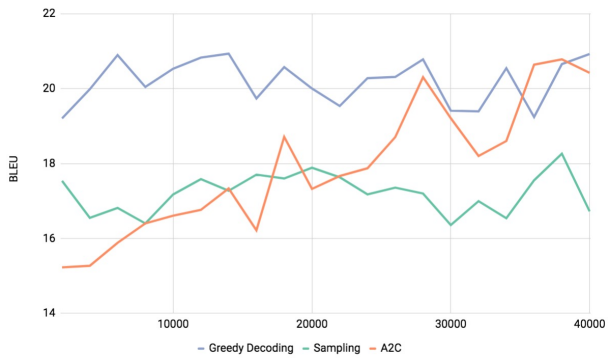

Figure 1: Comparing sampling, greedy decoding, and the A2C algorithm on the development data. Lines show average BLEU scores of every 2000 consecutive sentences.

peaky distribution and makes the model explore less.

It is best to have batching during bandit training for stability. Due to the limitation of the submission servers, that is, we only get the single reward feedback each time, we had to devise a method for batching for the feedback from the server. We cache the rewards until we reach the batch size, then do a batch update. However, due to some bugs in the implementation of this method, some sentences are not submitted in the correct order. And at some test points on the training server the scores are near or equal to zero.

In Figure 1 we present some results from the development server. We use a data selection model (200k in-domain data, 30% out-of-domain training data) as the baseline translation model, upon which we use the A2C algorithm to improve further. From this model, we generate translations with both sampling and greedy decoding to see how much the exploration required by the A2C algorithm hurts the performance. Figure 1 shows the average BLEU score of every 2000 sentences from the development server. A2C loses at the beginning because of exploration, and catches up as it sees more examples. Using sampling instead of greedy decoding, but exploration eventually improves the model.

6 Conclusion

We present the University of Maryland neural machine translation systems for the WMT17 bandit MT shared task. We employ two approaches: out-of-domain data selection and reinforcement learning. Experiments show that the best performance is achieved with a model pre-trained with only one-third of the available out-of-domain data. When applying reinforcement learning to further improve this model with bandit feedback, the model performance degrades initially due to exploration but gradually improves over time. Future work is to determine if reinforcement learning is more effective on a larger bandit learning dataset.

Acknowledgements

The authors thank the anonymous reviewers for many helpful comments. We would like to thank the task organizers: Pavel Danchenko, Hagen Fuerstenau, Julia Kreutzer, Stefan Riezler, Artem Sokolov, Kellen Sunderland, and Witold Szymaniak for organizing the task and for their help throughout the process.

This work was supported by NSF grants IIS-1320538 and IIS-1618193, as well as an Amazon Research Award and LTS grant DO-0032. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor(s).

References

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multi-armed bandit problem. *SIAM journal on computing* 32(1):48–77.

Alina Beygelzimer, Lihong Li, Robert E Schapire, John Langford, and Lev Reyzin. 2010. An optimal high probability algorithm for the contextual bandit problem. Technical report.

Kai-Wei Chang, He He, Hal Daumé III, and John Langford. 2015. Learning to search for dependencies. *arXiv preprint arXiv:1503.05615* .

Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. 2011. Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369* .

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 187–197.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810* .

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Association of Computational Linguistics*.

John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*. pages 817–824.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*. Association for Computational Linguistics, pages 220–224.

Herbert Robbins. 1985. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, Springer, pages 169–177.

Anthony Rousseau. 2013. Xenc: An open-source tool for data selection in natural language processing. *The Prague Bulletin of Mathematical Linguistics* (100):73–82.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016a. Learning structured predictors from bandit feedback for interactive nlp. ACL.

Artem Sokolov, Julia Kreutzer, Stefan Riezler, and Christopher Lo. 2016b. Stochastic structured prediction under bandit feedback. In *Advances in Neural Information Processing Systems*. pages 1489–1497.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.