Machine Translation System Selection from Bandit Feedback

Jason Naradowsky narad@preferred.jp
Preferred Networks, Tokyo, Japan

Xuan Zhang xuanzhang@jhu.edu
Kevin Duh kevinduh@cs.jhu.edu
Johns Hopkins University, Baltimore, USA

Abstract

Adapting machine translation systems in the real world is a difficult problem. In contrast to offline training, users cannot provide the type of fine-grained feedback (such as correct translations) typically used for improving the system. Moreover, different users have different translation needs, and even a single user's needs may change over time.

In this work we take a different approach, treating the problem of adaptation as one of selection. Instead of adapting a single system, we train many translation systems using different architectures, datasets, and optimization methods. Using bandit learning techniques on simulated user feedback, we learn a policy to choose which system to use for a particular translation task. We show that our approach can (1) quickly adapt to address domain changes in translation tasks, (2) outperform the single best system in mixed-domain translation tasks, and (3) make effective instance-specific decisions when using contextual bandit strategies.

1 Introduction

Recent advances in machine translation have greatly improved translation quality on in-domain data (Vaswani et al., 2017). But choosing the best system to deploy for a given translation task can be difficult, as many different systems could be considered approximately state-of-the-art, and there is not a single system which is best for all situations. For instance, while neural machine translation (NMT) traditionally excels in big data scenarios, when data is scarce it is not uncommon for a statistical phrased-based translation (SMT) to be the better choice. Even different hyperparameter settings of the same model may yield systems which each excel at different translations tasks.

In this work we explore the practical question of how best to deploy and improve an MT service over time. One solution to this problem is adaptation, where the model continues to train in an online manner during deployment, and the model parameters are updated in response to new types of data. However, adapting a system in this manner has the potential to cause catastrophic forgetting (Kirkpatrick et al., 2016): the model parameters shift too much, and performance on the original translation task declines.

A second practical concern is that for translation services deployed in the real world, the degree of feedback is often limited. Users of Google Translate can rate the quality of a translation as "helpful" or "wrong", and Facebook users can use an ordinal scale from 1 to 5, but neither can realistically ask a user to provide a reference translation. Furthermore, the user provides feedback only a single time per instance, and multiple users are unlikely to ask for

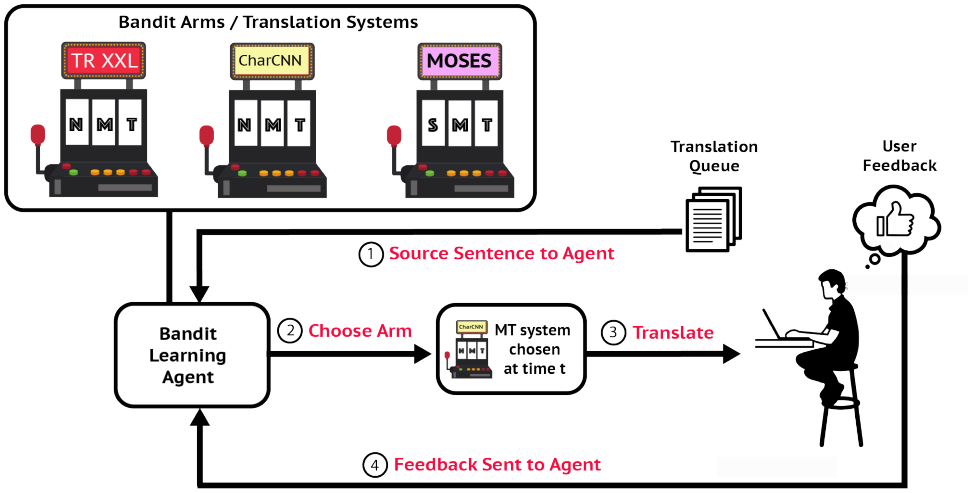

Figure 1: Basic setup: Bandit Learning Agent selects the MT system to use at time t. Based on user feedback (e.g. thumbs up/down or other signals), the bandit learning agent adapts to the stream of successive source sentences. The goal is to adapt quickly with few user feedback.

translations of the same sentence. To what extent can we leverage such feedback to quickly adapt our system to the user's translation needs?

Here we turn to *bandit learning*, a class of strategies for learning a decision policy in an online setting from such limited feedback. We assume access to a number of pre-trained machine translation systems, which vary in terms of architecture, training data, and optimization method. The policy must then determine which translation system to use for a given source sentence. In contrast to adaptation, the parameters of each translation system are fixed, preventing the possibility of catastrophic forgetting.

The bandit learning setup is illustrated in Figure 1. We assume no prior knowledge of the domain(s) of sentences that the user wants to translate. Source sentences are fed to the system in sequence, and at each time step the bandit learning agent chooses one pre-built MT system to generate translations. The user provides simple feedback for each translation (such as a thumbs up/down), and the goal of the agent is to converge to the best MT system as quickly as possible.

We compare several bandit methods across three data domains (and mixtures of these domains), and find that:

- Even simple bandit algorithms can quickly adapt to new domains, and converge to choosing optimal/near-optimal systems after a few hundred examples.

- For the case of contextual bandits, where we can condition on a particular source sentence when making a decision, simple features derived from sentence length, vocabulary, and BERT, are effective. In comparison to an oracle which chooses the single best arm for a given test set, our contextual system can vastly outperform it on mixed-domain settings.

- The methods are robust to different forms of simulated human feedback proposed in the machine translation literature.

We present bandit-based translation system selection as a viable alternative to deploying or adapting a single MT system.

2 Bandit Learning

We now provide a brief overview of bandit algorithms. Borrowing terminology from casino slot-machines, which are sometimes referred to as "one-armed bandits", a bandit problem presents the gambler with a choice: given a bandit with multiple arms, which should be pulled to maximize overall earnings? Assume that each arm has its own payoff distribution. Even though this distribution is not observed, the gambler may start preferring a particular arm over time, associating it with higher reward than others.

Formally, let there be a K-arm bandit, where each arm corresponds to an action. At each timestep t, an agent must choose an action $a_k, 1 \leq k \leq K$. Each arm is associated with a reward r_k^t, where higher values are desired. The agent's goal is to minimize cumulative regret (the amount of reward lost by making suboptimal decisions) over the course of T timesteps:

$$Regret = \sum_{t}^{T} (\max_{k'} \ r_{k'}^t) - r_\pi^t \tag{1}$$

where r_π^t denotes the reward of the arm chosen by the agent's policy. Note the agent does not necessarily minimize regret in the sense that it is an objective function to optimize. It does not have access to the oracle action nor the value $(\max_{k'} \ r_{k'}^t)$; instead, the agent receives information only on the arm it chooses (r_π^t). Intuitively, one can imagine that the agent needs to try different actions in order build up a profile of rewards for each arm.

Thus a K-arm bandit is a classic exploration vs. exploitation problem. Exploring new or infrequent actions improves our understanding of their effects, and may lead us to discover better strategies. However, doing so comes at the cost of not exploiting the actions we currently believe to be the best. There are many established algorithms for solving bandit problems, each approaching this trade-off in a different way. We introduce some of these methods later in Section 4.3, before applying them to the practical problem of simulated human-in-the-loop translation system selection.

3 Translation Bandits

We now define online translation system selection as a bandit learning problem. In our formulation, each arm is a translation system, i.e., pulling a bandit arm is equivalent to selecting a pre-trained translation system and applying it to a given source sentence. Learning takes place across a series of rounds, and our goal is to continually adapt the choice of translation system to changing user needs (represented by the source domain), thereby maximizing user satisfaction.

We assume that the source sentences that need to be translated are in a queue, and we process them in sequential order. The bandit selection process (at time t) is as follows:

1. Observe a source sentence s^t
2. (Optionally) Compute features $\phi(s^t)$
3. Choose a MT system $k^t = \pi(s^t)$ and generate a translation $g^t = k^t(s^t)$ for the user
4. The user gives feedback in the form of a reward r_π^t based on the quality of the translation. This can be, for example, a thumbs up/down rating. In our *simulation* experiments, we compute reward as $e^t = SentenceBLEU(s^t, g^t)$ based on reference translation g^t and perturb it to approximate coarse, noisy human feedback.[1]
5. Update selection policy π

[1] We compute sentence-level BLEU because rewards are defined per example for standard bandits. Nevertheless, our final evaluation is in terms of corpus-level BLEU. Note that we use perturbed SentenceBLEU only as a way to simulate human feedback; we do not assume sentence-level feedback to be consistent with corpus-level metrics.

Contextual Bandits One of the major distinctions between classes of bandit algorithms is whether the policy can condition its decision on a time-specific observation. Methods which do are referred to as *contextual bandits*, and are able to make more nuanced, instance-specific decisions. In other words, whereas simple bandits attempt to learn which arm is best, contextual bandits also learn *when* it is best. However, this necessitates computing a feature representation from the source sentence (Step 2).

Simulated Feedback In a real world deployment, user satisfaction is obtained directly from the user, but for the purpose of conducting this study in a tractable and repeatable manner, we simulate the human-in-the-loop. In this simulated setting we have access to reference translations, and can compute the true SentenceBLEU score (Step 4). However, using such a metric would not be a realistic approximation of real user feedback. Previous work (Nguyen et al., 2017) identified several ways in which human judgements may differ from more traditional continuous MT evaluation metrics:

- (1) *granularity* (thumbs-up vs. thumbs-down, or scoring on a 1-5 scale)
- (2) *variance* (different users may rate the same output differently)
- (3) *skew* (a user might be a harsher critic in general, and be prone to giving lower scores on average than other users).

In the experiments, we simulate use feedback by transforming raw SentenceBLEU scores into lower *granularity* bins on a 1-5 rating scale (Step 4). In Section 5.3 we further assess the effect of different simulated feedback to bandit learning.

4 Experiments

We aim to study the effectiveness of bandit algorithms on the task of MT system selection, across a variety of domains (and domain mixtures). Here we introduce the MT systems, the datasets used to train them, and the bandit algorithms used to learn the system selection policy.

4.1 Datasets

Our experiment data consists of three different tasks, translating from German to English:

1. The **General-Domain** task includes data from a range of domains, and is meant to be reflective of the kind of data used in public deployed systems. Specifically, we include OpenSubtitles2018 (Lison and Tiedemann, 2016) and WMT 2017 (Bojar et al., 2017), which contains data from e.g. parliamentary proceedings (Europarl, UN), political/economic news, and web-crawled parallel corpus (Common Crawl). After filtering out long sentences (>80 tokens), we obtain a training set of 28 million sentence pairs.

2. The **TED** task focuses on translating captions from TED Talks, which contains specialized vocabulary in various professional fields (e.g. technology, entertainment, design) in the form of monologue speeches. We use the WIT3 data distribution (Cettolo et al., 2012) with the train/dev/test splits provided by Duh (2018).

3. The **WIPO** task focuses on patent translation, which contains even more specialized jargon, written in a formal style. We use the COPPA V2.0 distribution (Junczys-Dowmunt et al., 2016). We held out 3000 random sentences each for dev and test, leaving 821 thousand sentences as training data.

	GENERAL		TED		WIPO		ALL	
	BLEU↑	TER↓	BLEU↑	TER↓	BLEU↑	TER↓	BLEU↑	TER↓
nmt-general	**29.4**	**50.5**	34.2	42.6	36.0	49.3	33.9	48.7
smt-general	23.9	54.9	30.7	45.8	26.7	56.9	26.5	53.8
smt-ted	16.5	62.3	28.7	47.9	12.0	69.5	15.6	61.6
nmt-ted	16.5	67.3	31.5	46.3	8.4	90.1	14.0	69.8
nmt-cont-ted	27.5	53.0	**39.3**	**38.2**	29.5	61.5	30.2	52.6
smt-wipo	9.9	79.3	9.7	77.5	51.2	36.0	35.2	66.6
nmt-wipo	6.6	101.2	7.7	92.1	61.9	25.4	39.0	77.8
nmt-cont-wipo	8.0	99.4	10.0	88.3	**62.3**	**25.0**	39.6	76.0

Table 1: Overview of translation performance (DE $\rightarrow$ EN) for the eight systems which constitute the arms of the bandit. Three architectures (nmt, nmt-cont, and smt) are trained and evaluated across three different domains. Typically NMT with continued training (nmt-cont) is the highest performing system on in-domain data, but other systems offer more consistent performance.

4.2 MT Systems

The training and development data described above are used to build machine translation systems. Bandit experiments are run on the test data, which has 1982, 3000, and 5504 sentences for the TED, WIPO, and General tasks respectively. All data is tokenized by the Moses tokenizer (Koehn et al., 2007), then split into subwords by BPE (Sennrich et al., 2016) independently with 30k merge operations for the English and German sides.

Neural machine translation Models are built with Sockeye (Hieber et al., 2017), using common settings for LSTM seq2seq models (Bahdanau et al., 2015): 2 layer encoder, 2 layer decoder, 512 hidden nodes, 512 word embedding sizes.

Statistical machine translation Models are built with Joshua (Post et al., 2013). This represents a strong phrase-based SMT model with GIZA++ alignments, 4-gram language model, and MIRA-based discriminative tuning.

Systems For each task, we train SMT and NMT models from scratch using only the training data in the respective domains, resulting in 6 models: {nmt,smt}-{general,ted,wipo}. Additionally we include two improved NMT models for WIPO and TED (nmt-cont-{ted,wipo}), which starts with nmt-general as initializaton and fine-tunes on WIPO or TED training data. This continued training process usually achieves strong translation performance in the target domain, but shows increased risk of catastrophic forgetting in the original general domain task (Luong and Manning, 2015; Thompson et al., 2019). This brings the total number of systems (also the number of bandit arms, K) to 8. The baseline performance of each system on the test sets in each domain is shown in Table 1.

Metrics The bandit learning agents are updated on-the-fly on the aforementioned test sets, using perturbed/granularized sentence-level BLEU as feedback. However, for final evaluation we collect all the resulting translations and compute corpus-level BLEU (Papineni et al., 2002) (implemented via SacreBLEU (Post, 2018)) and TER (Snover et al., 2006). We experiment with different ways to mix the test sets to illustrate different scenarios for bandit learning. For error analysis, we computed regret as in Eq. 1.

	GENERAL			TED			WIPO			AVG		
	R↓	B↑	T↓	R↓	B↑	T↓	R↓	B↑	T↓	R↓	B↑	T↓
random	16.9	17.9	70.9	20.8	24.7	59.3	32.2	36.7	52.4	23.3	26.4	60.9
best-arm-oracle	5.9	29.4	50.5	6.2	39.3	38.2	4.5	62.3	25.0	5.5	43.7	37.9
oracle	2.2	31.6	49.9	1.8	42.1	37.2	2.3	61.7	24.1	2.1	45.1	37.0
epsilon-greedy	9.3	26.2	56.9	11.9	34.1	45.2	12.6	54.8	32.6	11.3	38.3	44.9
ucb	9.9	25.4	56.7	13.2	32.7	46.8	9.3	58.0	29.8	10.8	38.7	44.4
linucb	**7.5**	**28.0**	**52.6**	**9.7**	**36.5**	**42.2**	**5.3**	**61.7**	**25.7**	**7.5**	**42.1**	**40.2**

Table 2: Results on the in-domain test sets. Evaluation is measured in terms of average regret (R), BLEU (B), and TER (T). Lower scores are better for regret and TER; higher scores are better for BLEU.

4.3 Bandit Methods

We compare the three bandit methods:

- **Epsilon-Greedy** The agent either exploits the arm with highest average reward with probability $1 - \epsilon$, or chooses randomly (uniformly) with probability ϵ (Sutton and Barto, 1998). Intuitively, the agent maintains a running average of each arm's rewards; it greedily chooses the one with the highest reward, but occasionally tries a random arm in order to improve its estimate of the running average.

- **Upper Confidence Bound (UCB)** An ϵ-greedy strategy is prone to obvious pitfalls. For instance, if the optimal action performs poorly early on, it may take many iterations to correct the agent's behavior. Alternatively, the agent can avoid becoming over-confident in its action reward estimates by establishing an upper confidence bound on each. This encourages the model to explore actions which may have low empirical estimates of reward, if they have been tried infrequently. The particular UCB algorithm we use here is UCB1 (Auer et al., 2002).

- **Lin-UCB** Recall that in contextual bandits, the learner is presented with a feature vector, here derived from the source sentence. The agent may use these feature vectors, along with the rewards of arms played in the past, to make a more informed choice of which arm to play at the current time step. Over time, the learner's aim is to collect enough information about how the context vectors and rewards relate to each other, so that it can predict the next best arm to play by looking at the feature vectors.

  LINUCB extends the principles of UCB to the contextual bandit scenario, using a linear model to predict the action (Li et al., 2010). We explore various features for this system (results in Sec. 5.4)

We also introduce three baseline systems: RANDOM, ORACLE, which always chooses the optimal translation system, and BEST-ARM-ORACLE, which chooses single best system (the one which has the highest BLEU across the entire test set).

5 Results

5.1 Overview: Comparison of Bandit Algorithms

We begin by assessing the relative strengths of bandit algorithms on the translation system selection task, and examine how closely performance on the regret-based objective function

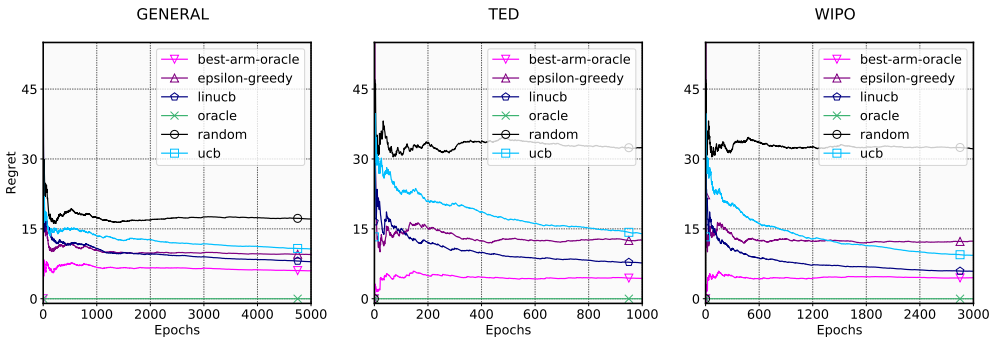

Figure 2: Comparison of cumulative regret in bandit algorithms across domains. The relative ordering of the algorithms is consistent across all three domains, with epsilon-greedy adapting early, before eventually being surpassed by linucb, which converges closer to the best-arm oracle. Note that cumulative regret lags behind changes in policy. Systems begin to find reasonable policies around epoch 50, marked by the sharp negative trajectory in cumulative regret.

corresponds to changes in translation evaluation measures like BLEU and TER. In these experiments we run each algorithm on the full test data for each of the three domains. As shown in Table 1, the highest-performing system for each domain is always an in-domain variant of neural translation. In this setting a good algorithm should quickly learn which arms are trained on in-domain data, and to exploit these as much as possible.

Fig. 2 illustrates the overall performance of EPSILON-GREEDY, UCB, and LINUCB, with respect to two oracles and a random baseline. We find that EPSILON-GREEDY is the fastest to converge. As the ϵ parameter balances exploration and exploitation, this behavior is somewhat within our control. Our results are obtained with $\epsilon = 0.3$. Empirically we observe that performance is quite robust to changes in ϵ, and values in the range 0.2 to 0.4 result in negligible performance differences (comparable to changing the random seed). In comparison, UCB performance follows a promising trajectory, but takes many more rounds to converge. For the goal of tailoring translations based on user feedback, executing thousands of interactions is likely an exorbitant requirement.

Turning to LINUCB, the contextual bandit algorithm, note the large gap between ORACLE and BEST-ARM-ORACLE performance. This is representative of the potential for additional performance gains when using a contextual method. Perhaps unsurprisingly then, we find that LINUCB typically outperforms other bandit algorithms in later epochs, often closely approximating the performance of the BEST-ARM-ORACLE. However, LINUCB also performs well in early epochs, and is often the best performing system after 100 epochs. Strong early performance means there may be little compromise to using a contextual approach like LINUCB, even in single domain translation tasks.

It is also worth pointing out the small discrepancy between the cumulative regret and BLEU. For instance, this is evident in the general domain results, where LINUCB has lower cumulative regret, but a lower BLEU score. While these two metrics are highly correlated in our experiments, there is a margin of disagreement and the ranking of systems can sometimes flip when comparing across these metrics.

5.2 Adapting to new Domains

A more realistic scenario may be a mixed-domain task, in which the user's translation needs are not fixed, but change over time. We simulate this scenario by mixing data from each of the

	R↓	B↑	T↓
random	24.6	31.4	46
best-arm-oracle	17.9	34.2	42
oracle	0.0	57.9	32
epsilon-greedy	20.5	34.4	43
ucb	22.8	33.8	44
linucb	**17.4**	**46.9**	**41**

Table 3: Performance on randomly shuffled data. All bandit systems are able to adapt to new domains quickly enough to achieve performance comparable to choosing the single best system, but the contextual bandit significantly outperforms it.

three domains in different ratios. The main question we want to ask is: can the bandit algorithms outperform the BEST-ARM-ORACLE system? Doing so would be represent a clear advantage over deploying any single system.

We present the performance of these systems in Table 3. The results show that as the data is increasingly mixed, the contextual bandit, LINUCB, significantly outperforms any single system. When data is completely shuffled, this amounts to a gain of more than 12 BLEU over the single best system, a relative improvement of over 37%. This is also true of simpler bandits, and EPSILON-GREEDY also outperforms the single-best system, but by a narrower margin.

Heatmaps of the algorithm decisions (Figure 3) provide some insight into the behavior of these systems. In fully randomized sequences, we observe that after 100-200 iterations of learning, LINUCB is able to closely mimic the behavior of the ORACLE system, ultimately converging to a similar distribution over decisions.[2] EPSILON-GREEDY converges to predominantly choosing the second-best arm as a safe bet, while UCB, which adjusts its policy more slowly, never exhibits clear decision trends.

Even in less mixed scenarios (3), top), results show that LINUCB is an effective system. As the domain changes between TED and WIPO, LINUCB closely tracks the ORACLE decisions, even within the first 50 iterations, making the system a promising option for real-world deployment.

5.3 Simulated Bandit Feedback

In order to ascertain how sensitive bandit translation system selection is to the nature of simulated feedback, we explore different types of of constructing feedback. Recall the ways in which human feedback may differ from continuous metrics as identified in Nguyen et al. (2017) are granularity, variance, and skew.

To simulate granular feedback, we bin SentenceBLEU scores into one of 5 equally-sized bins. For variance in the feedback score, we sample the score from a Gaussian (with a variance shrinking parameter of 1.0) that is updated after each evaluation. For skew, we simulate a harsher critic which biases the output towards scores in a lower range (a skew of 0.25).[3] For the sake of comparison we also include a scaling function, which simply scales the BLEU score into a suitable [0,1] loss range. This serves as an "oracle" of what performance we might expect if we were somehow able to ask users to provide a fine-grained score like BLEU.

Table 4 shows the effect of the four simulated feedback styles (GRANULAR, VARIANCE,

[2]From the ORACLE heat map, we see that all 8 arms/systems are chosen at some point; this shows that even though we may have prior knowledge that one system is generally better than another, it is still useful to include all systems if selection is performed at the sentence level.

[3]Experiments in the previous section used granular feedback method.

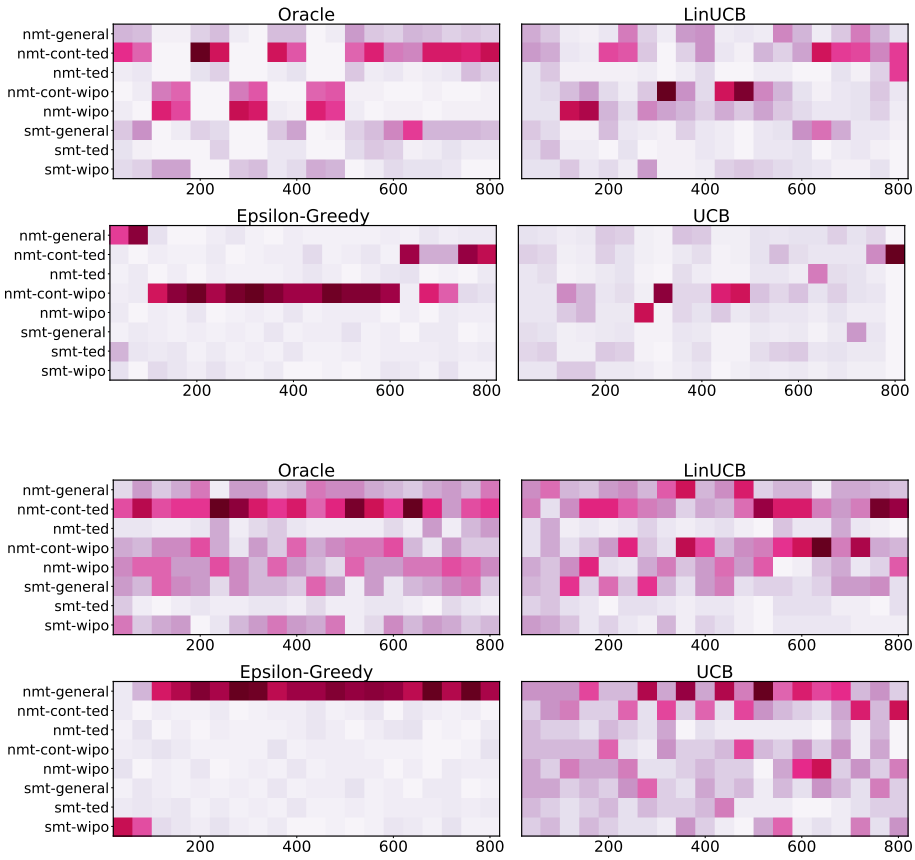

Figure 3: Decision heatmaps for shuffled data depicting the behavior of bandit algorithms across time. Each column of the heatmap represents the distribution over the choices made by the agent during that interval of training (red squares correspond to actions taken more frequently). In the top four plots we cycle the data domain every 100 examples. In the bottom four plots, the data is drawn randomly from each of the three domains. For clarity, we omit presenting heatmaps of the best-arm-oracle system, which is nmt-cont-ted in both scenarios.

SCALE, and SKEW) in a mixed data setting. We find that the nature of feedback has little overall effect on system performance. Altering the BLEU score to simulate SKEW was notably the most detrimental, but the remaining methods all performed similarly. Surprisingly we observe no significant difference between SCALE, which is essentially the full continuous BLEU metric, and other distortions to the feedback score.

5.4 Features for Contextual Bandits

Contextual bandit methods make use of feature vectors when choosing an action. In the context of translation, we can construct this vector from useful information in the source sentence. We experiment with three types of features:

- OOV, whether the source sentence contains a high proportion of out-of-vocabulary words,
- LEN, the length of the source sentence binned into five ranges of five (1-5, 6-10, ..., >25),[4]

[4]We use only the length feature for LINUCB in earlier in-domain experiments (Table 2).

	SCALE			VARIANCE			GRANULAR			SKEW		
	R↓	B↑	T↓	R↓	B↑	T↓	R↓	B↑	T↓	R↓	B↑	T↓
epsilon-greedy	19.3	35.0	42	21.6	37.5	44	19.0	37.3	42	19.7	33.1	43
ucb	13.3	50.1	38	13.3	50.7	38	13.1	50.2	38	12.3	51.0	38
linucb	14.1	48.9	39	13.5	48.9	39	13.6	48.5	39	15.6	45.2	40

Table 4: Effects of simulated feedback method on performance.

- BERT, features taken from a pre-trained language model (Devlin et al., 2019). Specifically, we ran the multilingual BERT base-size model out-of-the-box[5] in inference mode and extract the final layer of the transformer encoder. These embeddings are averaged across all tokens in the sentence. Since LINUCB has difficulty learning on large feature vectors, we only take the first 50 embedding dimensions as features.

A constant bias feature is used to establish a baseline. Table 5 shows the results. As evident by comparing against the BIAS feature results, all feature types provide useful information to the system selection task, though length and BERT features prove to be much more effective than vocabulary-based features.

	R↓	B↑	T↓
All	13.6	47.7	40
OOV	19.2	33.0	42
LEN	16.8	45.9	41
BERT	**13.3**	**48.1**	**40**
BIAS	19.0	31.8	42

Table 5: Ablation of contextual bandit features, on the randomly mixed-domain data.

6 Related Work

Multi-domain machine translation Our problem setting is closely related to multi-domain machine translation, where the data comes as a stream of sentences from a mixture of domains unknown to the model (Farajian et al., 2017; Huck et al., 2015). Typically using a single model, various extensions allow the system to cater more specifically to different types of source sentences. Such extensions may include data concatenation, model stacking, data selection and multi-model ensemble (Sajjad et al., 2017). One way is to pre-compute a domain label for each sentence using a dedicated classifier or model (Kobus et al., 2017; Tars and Fishel, 2018).

In neural models such adaptations can also be done in the learned representations. Britz et al. (2017) use a discriminator network on top of the encoder to distinguish between domains and pretend a domain token to the target sequence. Gu et al. (2019) employ a shared encoder-decoder and also private models to capture both domain-invariant and domain-specific knowledge, which are then combined to generate the target sequence. Such approaches can also be more nuanced, such as focusing on domain-specific words, and adjusting the training objective to emphasize their importance (Zeng et al., 2018).

A natural question is what advantage bandit system selection has over the alternative strategy of using of a domain classifier to determine which system to use. One advantage of bandit

[5] https://github.com/google-research/bert

selection is that it provides a unifying framework for online learning all aspects of problem. If a domain classifier is useful, the classifier's predictions can be used as a feature in the bandit algorithm, and the extent to which that feature is useful (or useful together with other information) will also be learned. Otherwise, the domain classifier features themselves can be incorporated into a contextual bandit policy where they can be access directly.

A second consideration is that the domains encountered in deployment may not be so well-defined and aligned to the translation systems as they are in our experimental setup. Imagine a specialized domain that is not similar to the training domain of any system. In these cases, other attributes of the model, such as its architecture or optimization strategy, may become more important. This is a situation the bandit approach is well-suited for.

Bandits in NLP Due to the high cost of sourcing human annotations for NLP tasks, developing tractable training methods for learning from simple feedback has long been a desirable goal. Learning NLP tasks ((machine translation, sequence labeling, text classification) from bandit feedback has been studied previously (Lawrence et al., 2017; Sokolov et al., 2016), and has been extended to train typical NLP architectures, such as neural sequence-to-sequence models (Kreutzer et al., 2017).

Bandits have also been applied previously in MT, the topic of a dedicated shared task (Sokolov et al., 2017). Within the context of bandit-driven MT, the focus has been on adapting an existing system, and are limited to simulated bandit feedback. Sokolov et al. (2016) used actual losses (BLEU) and pairwise ranking. Nguyen et al. (2017), also used bandit learning but to adapt a single neural MT system. In addition. Our approach is much different, in that it focuses on the use of a bandit-trained policy for selection, rather than adaptation or in-domain training.

7 Conclusion and Future Work

As MT systems become widely deployed, catering translation output to user needs, whether through adaptation or system selection, will become an increasingly important problem. In this work we showed that existing bandit algorithms are surprisingly effective at quickly adapting output to user needs, when the problem is phrased as one of selection. Contextual bandit system selection methods frequently outperform the use of a single translation system, establishing this technique as promising solution for dynamically adapting to user translation needs.

While we did not explore more recent bandit methods, including Bayesian bandits, or hierarchical bandits, intuitively such methods would be a good fit for a large scale version of this study. We assume arms are independent from one another, but they have dependencies both in terms of their architectures and in terms of their training data. One may also have prior knowledge of likely domains/arms, which can be incorporated in more advanced bandit methods.

We note that there are many kinds of deployment scenarios, depending on factors such as (a) the number of domains, (b) how much each domain drifts, and (c) whether feedback comes from a single user or multiple users. We have presented a proof-of-concept to show the potential effectiveness of bandits, but the exact scenarios where these methods are most appropriate still require more exploration. For example, the non-contextual bandit setup described here is appropriate for a computer-assisted translation application where one professional human translator post-edits sentences from long document translations. Here the number of post-edits can serves as implicit feedback, and bandit methods like EPSILON-GREEDY is likely to converge in time before the end of document for the translator to start seeing benefits.[6] On the other hand,

[6]An interesting extension is to collect the resulting post-edited translations in order to adapt a personalized MT engine, which can then be one of many arms in the bandit, or to allow multiple human translators to share their

the contextual bandit setup might be beneficial for an online MT service provider, where independent and unrelated translation requests may be interleaved. This can be viewed as another way to implement fast adaptation, but then one also needs to compare with the aforementioned multi-domain methods to decide the most suitable solution.

8 Acknowledgements

We thank the anonymous reviewers for their valuable comments. Artwork used in Fig.1 comes from TheNounProject[7], and the artists John Caserta, alifrio, and Herbert Spencer.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Proceedings of the international conference on learning representations (iclr). In *Neural Machine Translation by Jointly Learning to Align and Translate*.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Britz, D., Le, Q., and Pryzant, R. (2017). Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126, Copenhagen, Denmark. Association for Computational Linguistics.

Cettolo, M., Girardi, C., and Federico, M. (2012). Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Duh, K. (2018). The multitarget ted talks task. `http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/`.

Farajian, M. A., Turchi, M., Negri, M., and Federico, M. (2017). Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137, Copenhagen, Denmark. Association for Computational Linguistics.

Gu, S., Feng, Y., and Liu, Q. (2019). Improving domain adaptation translation with domain invariant and specific information. 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)*, pages 3081–3091, Minneapolis, Minnesota. Association for Computational Linguistics.

translation memories.

[7] `https://thenounproject.com/`

Hieber, F., Domhan, T., Denkowski, M., Vilar, D., Sokolov, A., Clifton, A., and Post, M. (2017). Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Huck, M., Birch, A., and Haddow, B. (2015). Mixed-domain vs. multi-domain statistical machine translation.

Junczys-Dowmunt, M., Pouliquen, B., and Mazenc, C. (2016). Coppa v2. 0: Corpus of parallel patent applications building large parallel corpora with gnu make. In *4th Workshop on Challenges in the Management of Large Corpora Workshop Programme*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. cite arxiv:1612.00796.

Kobus, C., Crego, J., and Senellart, J. (2017). Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 372–378, Varna, Bulgaria. INCOMA Ltd.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (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*.

Kreutzer, J., Sokolov, A., and Riezler, S. (2017). Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1503–1513. Association for Computational Linguistics.

Lawrence, C., Sokolov, A., and Riezler, S. (2017). Counterfactual learning from bandit feedback under deterministic logging : A case study in statistical machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2566–2576.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA. ACM.

Lison, P. and Tiedemann, J. (2016). OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

Luong, M.-T. and Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

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

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *ACL*.

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

Post, M., Ganitkevitch, J., Orland, L., Weese, J., Cao, Y., and Callison-Burch, C. (2013). Joshua 5.0: Sparser, better, faster, server. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 206–212, Sofia, Bulgaria. Association for Computational Linguistics.

Sajjad, H., Durrani, N., Dalvi, F., Belinkov, Y., and Vogel, S. (2017). Neural machine translation training in a multi-domain scenario. In *Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT)*.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *AMTA*.

Sokolov, A., Kreutzer, J., Lo, C., and Riezler, S. (2016). Learning structured predictors from bandit feedback for interactive NLP. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1610–1620, Berlin, Germany. Association for Computational Linguistics.

Sokolov, A., Kreutzer, J., Sunderland, K., Danchenko, P., Szymaniak, W., Fürstenau, H., and Riezler, S. (2017). A shared task on bandit learning for machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 514–524. Association for Computational Linguistics.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA.

Tars, S. and Fishel, M. (2018). Multi-domain neural machine translation. In *The 21st Annual Conference of the European Association for Machine Translation (EAMT)*, volume abs/1805.02282, Alacant, Spain.

Thompson, B., Gwinnup, J., Khayrallah, H., Duh, K., and Koehn, P. (2019). Overcoming catastrophic forgetting during domain adaptation of 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)*, pages 2062–2068, Minneapolis, Minnesota. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Neural Information Processing Systems (NIPS)*.

Zeng, J., Su, J., Wen, H., Liu, Y., Xie, J., Yin, Y., and Zhao, J. (2018). Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 447–457, Brussels, Belgium. Association for Computational Linguistics.