Learning to Actively Learn Neural Machine Translation

Ming Liu **Wray Buntine** **Gholamreza Haffari**
Faculty of Information Technology, Monash University
{ming.m.liu, wray.buntine, gholamreza.haffari} @ monash.edu

Abstract

Traditional active learning (AL) methods for machine translation (MT) rely on heuristics. However, these heuristics are limited when the characteristics of the MT problem change due to e.g. the language pair or the amount of the initial bitext. In this paper, we present a framework to *learn* sentence selection *strategies* for neural MT. We train the AL query strategy using a high-resource language-pair based on AL simulations, and then transfer it to the low-resource language-pair of interest. The learned query strategy capitalizes on the shared characteristics between the language pairs to make an effective use of the AL budget. Our experiments on three language-pairs confirms that our method is more effective than strong heuristic-based methods in various conditions, including cold-start and warm-start as well as small and extremely small data conditions.

1 Introduction

Parallel training bitext plays a key role in the quality neural machine translation (NMT). Learning high-quality NMT models in bilingually low-resource scenarios is one of the key challenges, as NMT's quality degrades severely in such setting (Koehn and Knowles, 2017).

Recently, the importance of learning NMT models in scarce parallel bitext scenarios has gained attention. Unsupervised approaches try to learn NMT models without the need for parallel bitext (Artetxe et al., 2017; Lample et al., 2017). Dual learning/backtranslation tries to start off from a small amount of bilingual text, and leverage monolingual text in the source and target language (Sennrich et al., 2015a; He et al., 2016). Zero/few shot approach attempts to transfer NMT learned from rich bilingual settings to low-resource settings (Johnson et al., 2016; Gu et al., 2018).

In this paper, we approach this problem from the active learning (AL) perspective. Assuming the availability of an annotation budget and a pool of monolingual source text as well as a small training bitext, the goal is to select the *most useful* source sentences and query their translation from an oracle up to the annotation budget. The queried sentences need to be selected carefully to get the value for the budget, i.e. get the highest improvements in the translation quality of the retrained model. The AL approach is orthogonal to the aforementioned approaches to bilingually low-resource NMT, and can be potentially combined with them.

We present a framework to *learn* the sentence selection *policy* most suitable and effective for the NMT task at hand. This is in contrast to the majority of work in AL-MT where hard-coded heuristics are used for query selection (Haffari and Sarkar, 2009; Bloodgood and Callison-Burch, 2010). More concretely, we learn the query policy based on a high-resource language-pair sharing similar characteristics with the low-resource language-pair of interest. After trained, the policy is applied to the language-pair of interest capitalising on the learned signals for effective query selection. We make use of imitation learning (IL) to train the query policy. Previous work has shown that the IL approach leads to more effective policy learning (Liu et al., 2018), compared to reinforcement learning (RL) (Fang et al., 2017) . Our proposed method effectively trains AL policies for *batch* queries needed for NMT, as opposed to the previous work on single query selection.

We conduct experiments on three language pairs Finnish-English, German-English, and Czech-English. Simulating low resource scenarios, we consider various settings, including cold-start and warm-start as well as small and extremely small data conditions. The experiments show the effectiveness and superiority of our policy query compared to strong baselines.

2 Learning to Actively Learn MT

Active learning is an iterative process: Firstly, a model is built using some initially available data. Then, the most worthwhile data points are selected from the unlabelled set for annotation by the oracle. The underlying model is then re-trained using the expanded labeled data. This process is then repeated until the budget is exhausted. The main challenge is how to identify and select the most beneficial unlabelled data points during the AL iterations.

The AL strategy can be learned by attempting to actively learn on tasks sampled from a distribution over the tasks (Bachman et al., 2017). We *simulate* the AL scenario on *instances* of a low-resource MT problem created using the bitext of the resource-rich language pair, where the translation of some part of the bitext is kept hidden. This allows to have an *automatic* oracle to reveal the translations of the queried sentences, resulting in an efficient way to quickly evaluate an AL strategy. Once the AL strategy is learned on simulations, it is then applied to real AL scenarios. The more related are the low-resource language-pair in the real scenario to those used to train the AL strategy, the more effective the AL strategy would be.

We are interested to train a translation model $m_{\boldsymbol{\phi}}$ which maps an input sentence from a source language $\boldsymbol{x} \in \mathcal{X}$ to its translation $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}$ in a target language, where $\mathcal{Y}_{\boldsymbol{x}}$ is the set of candidate translations for the input $\boldsymbol{x}$ and $\boldsymbol{\phi}$ is the parameter vector of the translation model. Let $D = \{(\boldsymbol{x}, \boldsymbol{y})\}$ be a support set of parallel corpus, which is randomly partitioned into parallel bitext D^{lab}, monolingual text D^{unl}, and evaluation D^{evl} datasets. Repeated random partitioning creates multiple instances of the AL problem.

3 Hierarchical MDP Formulation

A crucial difference of our setting to the previous work (Fang et al., 2017; Liu et al., 2018) is that the AL *agent* receives the reward from the oracle only after taking a *sequence* of actions, i.e. selection of an AL batch which may correspond to multiple training minibatches for the underlying NMT model. This fulfils the requirements for effective training of NMT, as minibatch updates are more effective than those of single sentence pairs. Furthermore, it is presumably more efficient and practical to query the translation of an untranslated batch from a human translator, rather than one sentence in each AL round.

At each time step t of an AL problem, the algorithm interacts with the oracle and queries the labels of a *batch* selected from the pool D_t^{unl} to form $\boldsymbol{b}_t$. As the result of this *sequence of actions* to select sentences in $\boldsymbol{b}_t$, the AL algorithm receives a reward $\text{BLEU}(m_{\boldsymbol{\phi}}, D^{evl})$ which is the BLEU score on D^{evl} based on the retrained NMT model using the batch $m_{\boldsymbol{\phi}}^{\boldsymbol{b}_t}$.

Formally, this results in a hierarchical Markov decision process (HMDP) for batch sentence selection in AL. A state $\boldsymbol{s}_t := \langle D_t^{lab}, D_t^{unl}, \boldsymbol{b}_t, \boldsymbol{\phi}_t \rangle$ of the HMDP in the time step t consist of the bitext D_t^{lab}, the monotext D_t^{unl}, the current text batch $\boldsymbol{b}_t$, and the parameters of the currently trained NMT model $\boldsymbol{\phi}_t$. The high-level MDP consists of a *goal* set $\mathcal{G} := \{retrain, halt_{\text{HI}}\}$, where setting a goal $g_t \in \mathcal{G}$ corresponds to either halting the AL process, or giving the execution to the low-level MDP to collect a new batch of bitext $\boldsymbol{b}_t$, re-training the underlying NMT model to get the update parameters $\boldsymbol{\phi}_{t+1}$, receiving the reward $R_{\text{HI}}(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) := \text{BLEU}(m_{\boldsymbol{\phi}_{t+1}}, D^{evl})$, and updating the new state as $\boldsymbol{s}_{t+1} = \langle D_t^{lab} \cup \boldsymbol{b}_t, D_t^{unl}, \emptyset, \boldsymbol{\phi}_{t+1} \rangle$. The $halt_{\text{HI}}$ goal is set in case the full AL annotation budget is exhausted, otherwise the re-train goal is set in the next time step.

The low-level MDP consists of primitive actions $a_t \in D_t^{unl} \cup \{halt_{\text{LO}}\}$ corresponding to either selecting of the monolingual sentences in D_t^{unl}, or halting the low-level policy and giving the execution back to the high-level MDP. The halt action is performed in case the maximum amount of source text is chosen for the current AL round, when the oracle is asked for the translation of the source sentences in the monolingual batch, which is then replaced by the resulting bitext. The sentence selection action, on the other hand, forms the next state by adding the chosen monolingual sentence to the batch and removing it from the pool of monolingual sentences. The underlying NMT model is not trained as a result of taking an action in the low-level policy, and the reward function is constant zero.

A trajectory in our HMDP consists of $\sigma := (\boldsymbol{s}_1, g_1, \tau_1, r_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_H, g_H, r_H, \boldsymbol{s}_{H+1})$ which is the concatenation of interleaved high-level trajectory $\tau_{HI} := (\boldsymbol{s}_1, g_1, r_1, \boldsymbol{s}_2, .., \boldsymbol{s}_{H+1})$ and low-level

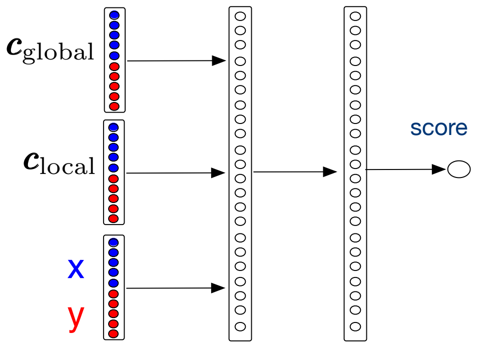

Figure 1: The policy network.

trajectories $\tau := (\boldsymbol{s}_1, a_1, \boldsymbol{s}_2, a_2, ..., \boldsymbol{s}_T, a_T, \boldsymbol{s}_{T+1})$. Clearly, the intermediate goals set by the top-level MDP into the σ are $retrain$, and only the last goal g_H is $halt_{\text{HI}}$, where H is determined by checking whether the total AL budget $\mathcal{B}_{\text{HI}}$ is exhausted. Likewise, the intermediate actions in τ_h are sentence selection, and only the last action a_T is $halt_{\text{LO}}$, where T is determined by checking whether the round-AL budget $\mathcal{B}_{\text{LO}}$ is exhausted.

We aim to find the optimal AL *policy* prescribing which datapoint needs to be queried in a given state to get the most benefit. The optimal policy is found by maximising the expected long-term reward, where the expectation is over the choice of the synthesised AL problems and other sources of randomness, i.e. partioing of D into D^{lab}, D^{unl}, and D^{evl}. Following (Bachman et al., 2017), we maximise the sum of the rewards after each AL *round* to encourage the *anytime* behaviour, i.e. the model should perform well after each batch query.

4 Deep Imitation learning for AL-NMT

The question remains of how to train the policy network to maximize the reward, i.e. the generalisation performance of the underlying NMT model. As the policy for the high-level MDP is fixed, we only need to learn the optimal policy for the low-level MDP. We formulate learning the AL policy as an imitation learning problem. More concretely, the policy is trained using an *algorithmic expert*, which can generate a *reasonable* AL trajectories (batches) for each AL state in the high-level MDP. The algorithmic expert's trajectories, i.e. sequences of AL states paired with the expert's actions in the low-level MDP, are then used to train the policy network. As such, the policy network is a classifier, conditioned on a context summarising both global and local histories, to choose the best sentence (action) among the candidates. After the AL policy is trained based on AL simulations, it is then transferred to the real AL scenario.

For simplicity of presentation, the training algorithms are presented using a fixed number of AL iterations for the high-level and low-level MDPs. This corresponds to AL with the sentence-based budget. However, extending them for AL with token-based budget is straightforward, and we experiment with both versions in §5.

Policy Network's Architecture The policy scoring network is a fully-connected network with two hidden layers (see Figure 1). The input involves the representation for three elements: (i) global context which includes all previous AL batches, (ii) local context which summarises the previous sentences selected for the current AL batch, and (iii) the candidate sentence paired with its translation generated by the currently trained NMT model.

For each source sentence $\boldsymbol{x}$ paired with its translation $\boldsymbol{y}$, we denote the representation by rep$(\boldsymbol{x}, \boldsymbol{y})$. We construct it by simply concatenating the representations of the source and target sentences, each of which is built by summing the embeddings of its words. We found this simple method to work well, compared to more complicated methods, e.g. taking the last hidden state of the decoder in the underlying NMT model. The global context ($\boldsymbol{c}_{\text{global}}$) and local contexts ($\boldsymbol{c}_{\text{local}}$) are constructed by summing the representation of the previously selected batches and sentence-pairs, respectively.

IL-based Training Algorithm The IL-based training method is presented in Algorithm 1. The policy network is initialised randomly, and trained based $\mathcal{T}$ simulated AL problems (lines 3–20), by portioning the available large bilingual corpus into three sets: (i) D^{lab} as the growing training bitex, (ii) D^{unl} as the pool of untranslated sentences where we pretend the translations are not given, and (iii) D^{evl} as the evaluation set used by our algorithmic expert.

For each simulated AL problem, Algorithm 1 executes T_{HI} iterations (lines 7–19) to collect AL batches for training the underlying NMT model *and* the policy network. An AL batch is obtained either from the policy network (line 15) or from the algorithmic expert (lines 10-13), depending on tossing a coin (line 9). The latter also includes adding the selected batch, the candidate batches, and the relevant state information to the *replay*

Algorithm 1 Learning AL-NMT Policy

```
Input: Parallel corpus $D$, $I_{\text{width}}$ the width of the constructed search lattices, the coin parameter $\alpha$, the number of sampled AL batches $K$
Output: policy $\pi$
1: $M \leftarrow \emptyset$                                   ▷ Replay Memory
2: Initialise $\pi$ with a random policy
3: for $\mathcal{T}$ training iterations do
4:     $D^{lab}, D^{evl}, D^{unl} \leftarrow$ randomPartition($D$)
5:     $\boldsymbol{\phi} \leftarrow$ trainModel($D^{lab}$)
6:     $\boldsymbol{c}_{\text{global}} \leftarrow \mathbf{0}$
7:     for $t \leftarrow 1$ to $T_{\text{HI}}$ do                 ▷ MDP$_{\text{HI}}$
8:         $\mathcal{S} \leftarrow$ searchLattice($D^{unl}, I_{\text{width}}$)
9:         if coinToss($\alpha$) = Head then
10:            $\boldsymbol{B} \leftarrow \{\text{samplePath}(\mathcal{S}, \boldsymbol{\phi}, \boldsymbol{c}_{\text{global}}, \pi, \beta)\}_1^K$
11:            $\boldsymbol{B} \leftarrow \boldsymbol{B} + \text{samplePath}(\mathcal{S}, \boldsymbol{\phi}, \boldsymbol{c}_{\text{global}}, \pi, 0)$
12:            $\boldsymbol{b} \leftarrow \arg\max_{\boldsymbol{b}' \in \boldsymbol{B}} \text{BLEU}(m_{\boldsymbol{\phi}}^{\boldsymbol{b}'}, D^{evl})$     ▷ expert
13:            $M \leftarrow M \cup \{(\boldsymbol{c}_{\text{global}}, \boldsymbol{\phi}, \boldsymbol{B}, \boldsymbol{b})\}$
14:        else
15:            $\boldsymbol{b} \leftarrow \text{samplePath}(\mathcal{S}, \boldsymbol{\phi}, \boldsymbol{c}_{\text{global}}, \pi, 0)$     ▷ policy
16:        $D^{lab} \leftarrow D^{lab} + \boldsymbol{b}$
17:        $D^{unl} \leftarrow D^{unl} - \{\boldsymbol{x} \text{ s.t. } (\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{b}\}$
18:        $\boldsymbol{\phi} \leftarrow$ retrainModel($\boldsymbol{\phi}, D^{lab}$)
19:        $\boldsymbol{c}_{\text{global}} \leftarrow \boldsymbol{c}_{\text{global}} \oplus \text{rep}(\boldsymbol{b})$
20:    $\pi \leftarrow$ updatePolicy($\pi, M, \boldsymbol{\phi}$)
21: return $\pi$
```

Algorithm 2 samplePath (selecting an AL batch)

```
Input: Search lattice $\mathcal{S}$, global context $\boldsymbol{c}_{\text{global}}$, policy $\pi$, perturbation probability $\beta$
Output: Selected AL batch $\boldsymbol{b}$
1: $\boldsymbol{b} \leftarrow \emptyset$
2: $\boldsymbol{c}_{\text{local}} \leftarrow \mathbf{0}$
3: for $t \leftarrow 1$ to $T_{\text{LO}}$ do                    ▷ MDP$_{\text{LO}}$
4:     if coinToss($\beta$) = Head then
5:         $\boldsymbol{x}_t \leftarrow \pi_0(\mathcal{S}[t])$               ▷ perturbation policy
6:     else
7:         $\boldsymbol{x}_t \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{S}[t]} \pi(\boldsymbol{c}_{\text{global}}, \boldsymbol{c}_{\text{local}}, \boldsymbol{x})$
8:     $\boldsymbol{y}_t \leftarrow$ oracle($\boldsymbol{x}_t$)          ▷ getting the gold translation
9:     $\boldsymbol{c}_{\text{local}} \leftarrow \boldsymbol{c}_{\text{local}} \oplus \text{rep}(\boldsymbol{x}_t, \boldsymbol{y}_t)$
10:    $\boldsymbol{b} \leftarrow \boldsymbol{b} + (\boldsymbol{x}_t, \boldsymbol{y}_t)$
11: return $\boldsymbol{b}$
```

memory M, based on which the policy will be retrained. The selected batch is then used to retrain the underlying NMT model, update the training bilingual corpus and pool of monotext, and update the global context vector (lines 16–19).

The mixture of the policy network and algorithmic expert in batch collection on simulated AL problems is inspired by Dataset Aggregation DAGGER (Ross and Bagnell, 2014). This makes sure that the collected states-actions pairs in the replay memory include situations encountered beyond executing only the algorithmic expert. This informs the trained AL policy how to act reasonably in new situations encountered in the test time, where only the network policy is in charge and the expert does not exist.

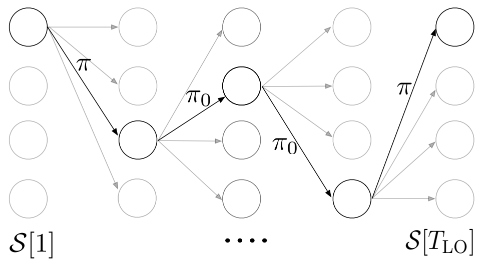

Figure 2: The search lattice and the selection of a batch based on the perturbed policy. Each circle denotes a sentence from the pool of monotext. The number of sentences at each time step, denoted by $\mathcal{S}[t]$, is I_{width}. The black sentences are selected in this AL batch.

Algorithmic Expert At a given AL state, the algorithmic expert selects a reasonable batch from the pool, D^{unl} via:

$$\arg\max_{\boldsymbol{b} \in \boldsymbol{B}} \text{BLEU}(m_{\boldsymbol{\phi}}^{\boldsymbol{b}}, D^{evl}) \quad (1)$$

where $m_{\boldsymbol{\phi}}^{\boldsymbol{b}}$ denotes the underlying NMT model ϕ further retrained by incorporating the batch $\boldsymbol{b}$, and $\boldsymbol{B}$ denotes the possible batches from D^{unl} (Liu et al., 2018). However, the number of possible batches is exponential in the size D^{unl}, hence the above optimisation procedure would be very slow even for a moderately-sized pool.

We construct a *search lattice* $\mathcal{S}$ from which the candidate batches in $\boldsymbol{B}$ are sampled (see Figure 2). The search lattice is constructed by sampling a fixed number of candidate sentences I_{width} from D^{unl} for each position in a batch, whose size is T_{LO}. A candidate AL batch is then be selected using Algorithm 2. It executes a mixture of the current AL policy π and a perturbation policy π_0 (e.g. random sentence selection or any other heuristic) in the lower-level MDP to sample a batch. After several such batches are sampled to form $\boldsymbol{B}$, the best one is selected according to eqn 1.

We have carefully designed the search space to be able to incorporate the current policy's recommended batch and sampled deviations from it in $\boldsymbol{B}$. This is inspired by the LOLS (Locally Optimal Learning to Search) algorithm (Chang et al., 2015), to invest efforts in the neighbourhood of the current policy and improve it. Moreover, having to deal with only I_{width} number of sentences at each selection stage makes the batch formation algorithm based on the policy fast and efficient.

Re-training the Policy Network To train our policy network, we turn sentence preference scores to probabilities over the candidate batches,

Algorithm 3 Policy Transfer

```
Input: Bitext $D^{lab}$, monotext $D^{unl}$, pre-trained policy $\pi$
Output: NMT model $\phi$
 1: Initialise $\phi$ with $D^{lab}$
 2: $c_{\text{global}} \leftarrow \mathbf{0}$
 3: for $t \leftarrow 1$ to $T_{HI}$ do
 4:     $\mathcal{S} \leftarrow$ searchLattice$(D^{unl}, I_{\text{width}})$
 5:     $\boldsymbol{b} \leftarrow$ samplePath$(\mathcal{S}, \phi, c_{\text{global}}, \pi, 0)$
 6:     $D^{lab} \leftarrow D^{lab} + \boldsymbol{b}$
 7:     $D^{unl} \leftarrow D^{unl} - \{\boldsymbol{x}$ s.t. $(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{b}\}$
 8:     $\phi \leftarrow$ retrainModel$(\phi, D^{lab})$
 9:     $D^{lab} \leftarrow D^{lab} + \boldsymbol{b}$
10:     $c_{\text{global}} \leftarrow c_{\text{global}} \oplus$ rep$(\boldsymbol{b})$
11: return $\phi$
```

and optimise the parameters to maximise the log-likelihood objective (Liu et al., 2018).

More specifically, let $(c_{\text{global}}, \boldsymbol{B}, \boldsymbol{b})$ be a training tuple in the replay memory. We define the probability of the correct action/batch as

$$Pr(\boldsymbol{b}|\boldsymbol{B}, c_{\text{global}}) := \frac{\text{score}(\boldsymbol{b}, c_{\text{global}})}{\sum_{\boldsymbol{b}' \in \boldsymbol{B}} \text{score}(\boldsymbol{b}', c_{\text{global}})}.$$

The preference score for a batch is the sum of its sentences' preference scores,

$$\text{score}(\boldsymbol{b}, c_{\text{global}}) := \sum_{t=1}^{|\boldsymbol{b}|} \pi(c_{\text{global}}, c_{\text{local}<t}, \text{rep}(\boldsymbol{x}_t, \boldsymbol{y}_t))$$

where $c_{\text{local}<t}$ denotes the local context up to the sentence t in the batch.

To form the log-likelihood, we use recent tuples and randomly sample several older ones from the replay memory. We then use stochastic gradient descent (SGD) to maximise the training objective, where the gradient of the network parameters are calculated using the backpropagation algorithm.

Transfering the Policy We now apply the policy learnt on the source language pair to AL in the target task (see Algorithm 3). To enable transferring the policy to a new language pair, we make use of pre-trained multilingual word embeddings. In our experiments, we either use the pre-trained word embeddings from (Ammar et al., 2016) or build it based on the available bitext and monotext in the source and target language (c.f. §5.2). To retrain our NMT model, we make parameter updates based on the mini-batches from the AL batch as well as sampled mini-batches from the previous iterations.

5 Experiment

Datasets Our experiments use the following language pairs in the news domain based on WMT2018: English-Czech (EN-CS), English-German (EN-DE), English-Finnish (EN-FI). For AL evaluation, we randomly sample 500K sentence pairs from the parallel corpora in WMT2018 for each of the three language pairs, and take 100K as the initially available bitext and the rest of 400K as the pool of untranslated sentences, pretending the translation is not available. During the AL iterations, the translation is revealed for the queried source sentences in order to retrain the underlying NMT model.

For pre-processing the text, we normalise the punctuations and tokenise using `moses`[1] scripts. The trained models are evaluated using BLEU on tokenised and cased sensitive test data from the `newstest 2017`.

NMT Model Our baseline model consists of a 2-layer bi-directional LSTM encoder with an embeddings size of 512 and a hidden size of 512. The 1-layer LSTM decoder with 512 hidden units uses an attention network with 128 hidden units. We use a multiplicative-style attention attention architecture (Luong et al., 2015). The model is optimized using Adam (Kingma and Ba, 2014) with a learning rate of 0.0001, where the dropout rate is set to 0.3. We set the mini-batch size to 200 and the maximum sentence length to 50. We train the base NMT models for 5 epochs on the initially available bitext, as the perplexity on the dev set do not improve beyond more training epochs. After getting new translated text in each AL iteration, we further sample $\times 5$ more bilingual sentences from the previously available bitext, and make one pass over this data to re-train the underlying NMT model. For decoding, we use beam-search with the beam size of 3.

Selection Strategies We compare our policy-based sentence selection for NMT-AL with the following heuristics:

- **Random** We randomly select monolingual sentences up to the AL budget.
- **Length-based** We use shortest/longest monolingual sentences up to the AL budget.
- **Total Token Entropy (TTE)** We sort monolingual sentences based on their TTE which has been shown to be a strong AL heuristic (Settles and Craven, 2008) for

[1]http://www.statmt.org/moses

System	EN→DE	EN→FI	EN→DE	EN→CS	EN→CS	EN→FI
Base NMT (100K)	13.2	10.3	13.2	8.1	8.1	10.3
AL with 135K token budget						
Random	13.9	11.2	13.9	8.3	8.3	11.2
Shortest	14.5	11.5	14.5	**8.6**	8.6	11.5
Longest	14.1	11.3	14.1	8.2	8.2	11.3
TTE	14.2	11.3	14.2	8.5	8.5	11.3
Token Policy	**15.5**	**12.8**	**14.8**	8.5	**9.0**	**12.5**
AL with 677K token budget						
Random	15.9	13.5	15.9	**9.2**	9.2	13.5
Shortest	15.8	13.7	15.8	8.9	8.9	13.7
Longest	15.6	13.5	15.6	8.5	8.5	13.5
TTE	15.6	13.7	15.6	8.6	8.6	13.7
Token Policy	**16.6**	**14.1**	**16.3**	9.2	**10.3**	**13.9**
FULL bitext (500K)	20.5	18.3	20.5	12.1	12.1	18.3

Table 1: BLEU scores on tests sets with different selection strategies, the budget is at token level with annotation for 135.45k tokens and 677.25k tokens respectively.

sequence-prediction tasks. Given a monolingual sentence $\boldsymbol{x}$, we compute the TTE as $\sum_{i=1}^{|\hat{\boldsymbol{y}}|} \text{Entropy}[P_i(.|\hat{\boldsymbol{y}}_{<i}, \boldsymbol{x}, \boldsymbol{\phi})]$ where $\hat{\boldsymbol{y}}$ is the decoded translation based on the current underlying NMT model $\boldsymbol{\phi}$, and $P_i(.|\hat{\boldsymbol{y}}_{<i}, \boldsymbol{x}, \boldsymbol{\phi})$ is the distribution over the vocabulary words for the position i of the translation given the source sentence and the previously generated words. We also experimented with the normalised version of this measure, i.e. dividing TTE by $|\hat{\boldsymbol{y}}|$, and found that their difference is negligible. So we only report TTE results.

5.1 Translating from English

Setting We train the AL policy on a language-pair treating it as high-resource, and apply it to another language-pair treated as low-resource. To transfer the policies across languages, we make use of pre-trained multilingual word embeddings learned from monolingual text and bilingual dictionaries (Ammar et al., 2016). Furthermore, we use these cross-lingual word embeddings to initialise the embedding table of the NMT in the low-resource language-pair. The source and target vocabularies for the NMT model in the low-resource scenario are constructed using the initially available 100K bitext, and are expanded during the AL iterations as more translated text becomes available.

Results Table 1 shows the results. The experiments are performed with two limits on token annotation budget: 135k and 677k corresponding to select roughly 10K and 50K sentences in total in AL[2], respectively. The number of AL iterations is 50, hence the token annotation budget for each round is 2.7K and 13.5K. As we can see our policy-based AL method is very effective, and outperforms the strong AL baselines in all cases except, when transferring the policy trained on EN $\rightarrow$ FI to EN $\rightarrow$ CS where it is on-par with the best baseline.

System	EN→DE	EN→FI
Base NMT (100K)	13.2	10.3
AL with 10K sentence budget		
Random	14.2	11.3
Shortest	13.9	11.0
Longest	14.7	**11.8**
TTE	14.3	11.2
Sent $\pi_{\text{EN}\rightarrow\text{CS}}$	14.5	11.5
Token $\pi_{\text{EN}\rightarrow\text{CS}}$	**15.1**	**11.8**
AL with 50K sentence budget		
Random	16.1	13.5
Shortest	15.5	12.9
Longest	**17.2**	**14.2**
TTE	16.6	13.5
Sent $\pi_{\text{EN}\rightarrow\text{CS}}$	16.3	14.0
Token $\pi_{\text{EN}\rightarrow\text{CS}}$	16.8	14.1

Table 2: BLEU scores on tests sets for different language pairs with different selection strategies, the budget is at sentence level with annotation for 10k sentences and 50k sentences respectively.

Sentence vs Token Budget In our results in Table 1, we have taken the number of tokens in the selected sentences as a proxy for the annotation cost. Another option to measure the annotation cost is the number of selected sentences, which admittedly is not the best proxy. Nonetheless, one

[2]These token limit budgets are calculated using random selection of 10K and 50K sentences multiple times, and taking the average of the tokens across the sampled sets.

	100K initial bitext		10K initial bitext	
AL method	cold-start	warm-start	cold-start	warm-start
Base NMT	10.6/11.8	13.9/14.7	2.3/2.5	5.4/5.8
Random	12.9/13.3	15.1/16.2	5.5/5.6	9.3/9.6
Shortest	13.0/13.5	15.9/16.4	5.9/6.1	9.1/9.3
Longest	12.5/12.9	15.3/15.8	5.7/5.9	9.8/10.2
TTE	12.8/13.2	15.8/16.1	5.9/6.2	9.8/10.1
$\pi_{\text{CS}\to\text{EN}}$	13.9/**14.2**	16.8/**17.3**	6.3/**6.5**	10.5/**10.9**
$\pi_{\text{FI}\to\text{EN}}$	13.5/13.8	16.5/16.9	6.1/6.2	10.2/10.3
$\pi_{\text{EN}\to\text{CS}}$	13.3/13.6	16.4/16.5	5.1/5.7	10.3/10.5
$\pi_{\text{EN}\to\text{FI}}$	13.2/13.5	15.9/16.3	5.1/5.6	9.8/10.2
Ensemble				
$\pi_{\text{CS,FI}\to\text{EN}}$	14.1/**14.3**	16.8/**17.5**	6.3/**6.5**	10.5/**10.9**
$\pi_{\text{EN}\to\text{CS,FI}}$	13.6/13.8	16.5/16.9	5.8/5.9	10.3/10.5
Full Model (500K)	20.5/20.6	22.3/22.5	-	-

Table 3: BLEU scores on tests sets using different selection strategies. The token level annotation budget is 677K.

may be interested to see how different AL methods compare against each other based on this cost measure.

Table 2 show the results based on the sentence-based annotation cost. We train a policy on EN $\to$ CS, and apply it to EN $\to$ DE and EN $\to$ FI translation tasks. In addition to the token-based AL policy from Table 1, we *train* another policy based on the sentence budget. The token-based policy is competitive in EN $\to$ DE, where the longest sentence heuristic achieves the best performance, presumably due to the enormous training signal obtained by translation of long sentences. The token-based policy is on par with longest sentence heuristic in EN $\to$ FI for both 10K and 100K AL budgets to outperform the other methods.

5.2 Translating into English

Setting We investigate the performance of the AL methods on DE $\to$ EN based on the policies trained on the other language pairs. In addition to 100K training data condition, we assess the effectiveness of the AL methods in an extremely low-resource condition consisting of only 10K bilingual sentences as the initial bitext.

In addition to the source word embedding table that we initialised in the previous section's experiments using the cross lingual word embeddings, we are able further to initialise *all* of the other NMT parameters for DE $\to$ EN translation. This includes the target word embedding table and the decoder softmax, as the target language is the same (EN) in the language-pairs used for both policy training and policy testing. We refer to this setting as *warm-start*, as opposed to *cold-start* in which we only initialised the source embedding table with the cross-lingual embeddings. For the warm-start experiments, we transfer the NMT trained on 500K CS-EN bitext, based on which the policy is trained. We use byte-pair encoding (BPE) (Sennrich et al., 2015b) with 30K operations to bpe the EN side. For the source side, we use words in order to use the cross-lingual word embeddings. All parameters of the transferred NMT are frozen, except the ones corresponding to the bidirectional RNN encoder and the source word embedding table.

To make this experimental condition as realistic as possible, we learn the cross-lingual word embedding for DE using large amounts of monolingual text and the initially available bitext, assuming a multilingual word embedding already exists for the languages used in the policy training phase. More concretely, we sample 5M DE text from WMT2018 data[3], and train monolingual word embeddings as part of a skip-gram language model using `fastText`.[4] We then create a bilingual EN-DE word dictionary based on the initially available bitext (either 100K or 10K) using word alignments generated by `fast_align`.[5] The bilingual dictionary is used to project the monolingual DE word embedding space into that of EN,

[3]We make sure that it does not include the DE sentences in the 400K pool used in the AL experiments.

[4]https://github.com/facebookresearch/fastText

[5]https://github.com/clab/fast_align

hence aligning the spaces through the following orthogonal projection:

$$\arg\max_{\boldsymbol{Q}} \sum_{i=1}^{m} \boldsymbol{e}[y_i]^T \cdot \boldsymbol{Q} \cdot \boldsymbol{e}[x_i] \quad \text{s.t.} \quad \boldsymbol{Q}^T \cdot \boldsymbol{Q} = \boldsymbol{I}$$

where $\{(y_i, x_i)\}_{i=1}^m$ is the bilingual dictionary consisting of pairs of DE-EN words[6], $\boldsymbol{e}[y_i]$ and $\boldsymbol{e}[x_i]$ are the embeddings of the DE and EN words, and $\boldsymbol{Q}$ is the orthogonal transformation matrix aligning the two embedding spaces. We solve the above optimisation problem using SVD as in (Smith et al., 2017). The cross-lingual word embedding for a DE word y is then $\boldsymbol{e}[y]^T \cdot \boldsymbol{Q}$. We build two such cross-lingual embeddings based on the two bilingual dictionaries constructed from the 10K and 100K bitext, in order to use in their corresponding experiments.

Results Table 3 presents the results, on two conditions of 100K and 10K initial bilingual sentences. For each of these data conditions, we experiments with both cold-start and warm-start settings using the pre-trained multilingual word embeddings from (Ammar et al., 2016) or those we have trained with the available bitext plus additional monotext. Firstly, the warm start strategy to transfer the NMT system from CS $\rightarrow$ EN to DE $\rightarrow$ EN has been very effective, particularly on extremely low bilingual condition of 10K sentence pairs. It is worth noting that our multilingual word embeddings are very effective, even-though they are trained using small bitext. Secondly, our policy-based AL methods are more effective than the baseline methods and lead to up to +1 BLEU score improvements.

We further take the ensemble of multiple trained policies to build a new AL query strategy. In the ensemble, we rank sentences based on each of the policies. Then we produce a final ranking by combining these rankings. Specifically, we sum the ranking of each sentence according to each policy to get a rank score, and re-rank the sentences according to their rank score. Table 3 shows that ensembling is helpful, but does not produce significant improvements compared to the best policy.

5.3 Analysis

TTE is a competitive heuristic-based strategy, as shown in the above experiments. We compare

[6]One can incorporate human curated bilingual lexicons to the automatically curated dictionaries as well.

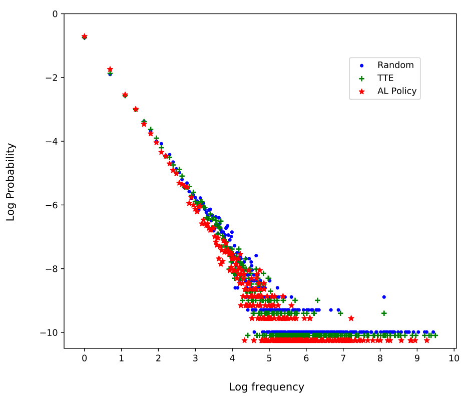

Figure 3: On the task of DE→EN, the plot shows the log fraction of words vs the log frequency from the selected data returned by different strategies, in which we have a 677K token budget and do warm start with 100K initial bitext. The AL policy here is $\pi_{EN\rightarrow CS}$.

the word frequency distributions of the selected source text returned by Random, TTE against our AL policy. The policy we use here is $\pi_{EN\rightarrow CS}$ and applied on the task of DE→EN, which is conducted in the warm-start scenario with 100K initial bitext and 677K token budget.

Fig. 3 is the log-log plot of the fraction of vocabulary words (y axis) having a particular frequency (x axis). Our AL policy is less likely to select high-frequency words than other two methods when it is given a fixed token budget.

6 Conclusion

We have introduced an effective approach for learning active learning policies for NMT, where the learner needs to make batch queries. We have provides a hierarchical MDP formulation of the problem, and proposed a policy network structure capturing the context in both MDP levels. Our policy training method uses imitation learning and a search lattice to carefully collect AL trajectories for further improvement of the current policy. We have provided experimental results on three language pairs, where the policies are transferred across languages using multilingual word embeddings. Our experiments confirms that our method is more effective than strong heuristic-based methods in various conditions, including cold-start and warm-start as well as small and extremely small data conditions.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. 2017. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 301–310, International Convention Centre, Sydney, Australia. PMLR.

Michael Bloodgood and Chris Callison-Burch. 2010. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 854–864.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé, III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*, pages 2058–2066.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018. Universal neural machine translation for extremely low resource languages. *arXiv preprint arXiv:1802.05368*.

Gholamreza Haffari and Anoop Sarkar. 2009. Active learning for multilingual statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL*, pages 181–189.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Vigas, Martin Wattenberg, G.s Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

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

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.