SURF: Semantic-level Unsupervised Reward Function for Machine Translation

Atijit Anuchitanukul[1], **Julia Ive**[2]
Imperial College London[1], Queen Mary University of London[2]
atijit.anuchitanukul20@imperial.ac.uk
j.ive@qmul.ac.uk

Abstract

The performance of Reinforcement Learning (RL) for natural language tasks including Machine Translation (MT) is crucially dependent on the reward formulation. This is due to the intrinsic difficulty of the task in the high-dimensional discrete action space as well as the sparseness of the standard reward functions defined for limited set of ground-truth sequences biased towards singular lexical choices. To address this issue, we formulate SURF, a maximally dense semantic-level unsupervised reward function which mimics human evaluation by considering both sentence fluency and semantic similarity. We demonstrate the strong potential of SURF to leverage a family of Actor-Critic Transformer-based Architectures with synchronous and asynchronous multi-agent variants. To tackle the problem of large action-state spaces, each agent is equipped with unique exploration strategies, promoting diversity during its exploration of the hypothesis space. When BLEU scores are compared, our dense unsupervised reward outperforms the standard sparse reward by 2% on average for in- and out-of-domain settings.

1 Introduction

Reinforcement Learning (RL) has shown promise in the field of text generation. This is mainly due to the fact that it allows the usage of non-differentiable evaluation functions fit for the discrete natural language tasks. It also serves as a solution for bridging the gap between training and inference time regimes ("exposure bias") that arises from the fact that the model is never exposed to its own errors as only ground-truth labels are used to condition the generation during training (Wang and Sennrich, 2020). One of the essential components of the RL framework is the reward function, which is used to provide agents with indicative signals in terms of the effectiveness of chosen actions.

The usage of RL in Neural Machine Translation (NMT) and language generation however has been doubted largely due to the difficulty of exploration in the high-dimensional discrete action space combined with the sparse reward signal. The latter comes from the typical metrics used as rewards (e.g., BLEU (Papineni et al., 2002)). These rewards evaluate text in a shallow way by measuring the string similarity between generated and ground-truth sequences, making them extremely sparse and biased towards singular lexical choices. As the RL policy is usually initialised with some pre-trained distribution over words, suspicion has been raised that in this situation, those words already most likely gain probability mass regardless of the rewards (Choshen et al., 2020). Thus, the current sparse rewards are not beneficial for rigorous exploration of different words during training. Recent studies suggest that the main benefit for NMT from RL is in performing domain adaptation when using proper hypothesis space exploration along with special emphasis on reward scaling and normalisation (Kiegeland and Kreutzer, 2021).

To address the problem posed by sparse or biased rewards, we propose SURF, a formulation of the unsupervised reward function that evaluates machine-generated texts in the semantic space by factoring in different qualitative aspects. Furthermore, we introduce an additional scaling and normalisation mechanism which ensures fairness and uniformity of the reward function regardless of the complexity of the natural language task.

Our **main contributions** are thus threefold: (a) the proposal of SURF, an unsupervised dense reward assessing both sentence fluency and adequacy (Section 4). We demonstrate this reward leads to a translation quality favourably comparable to the traditional sparse BLEU reward both in automatic and human evaluation; (b) the proposal of an additional normalisation using reward shaping mechanisms for the unsupervised reward; (c) demonstra-

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4508 - 4522

tion of the strong potential of the proposed reward to elicit benefits of various RL architectures. We experiment with multi-agent synchronous and asynchronous Actor-Critic (AC) architectures as applied to the problem of MT (Section 3.3). Each of the parallel agents in those architectures is trained using different segments of the training dataset which has their own unique exploration strategy. When BLEU scores are compared, our dense unsupervised reward outperforms the standard sparse reward by 2% on average for both in- and out-of-domain settings. To the best of our knowledge, this formulation of the unsupervised reward for a range of multi-agent architectures is the first of its kind for MT.

Our datasets and settings are described in Section 5, and results of our experiments are described in Section 6.

2 Related Work

The following section describes the work related to ours in the subfields of Machine Translation and RL for language generation.

Reinforcement Learning Algorithms for Neural Machine Translation REINFORCE (Williams, 1992) algorithm and its variants have so far been the most widely used RL algorithms in MT (Ranzato et al., 2015; Rennie et al., 2017; Paulus et al., 2018; Hu et al., 2018). The fact that REINFORCE-based approaches suffer from high variance in general and in MT in particular has stimulated attempts to apply Actor-Critic (AC) models to the task. The first attempt of the kind was the one of Bahdanau et al. (2016). More advanced AC models with Q-Learning are rarely applied to language generation problems. However, there are exceptions (e.g., entropy-regularised AC models that promote exploration of actions (Dai et al., 2018; Ive et al., 2021)). This could be explained by the difficulty of approximating the Q-function for large action space. In this work we explore a series of multi-agent AC architectures which to the best of our knowledge have never been applied to MT before.

Unsupervised Rewards for Language Generation Tasks Recent work on unsupervised rewards in NLP has explored both dynamic (Ive et al., 2021) and static rewards (Gao et al., 2020; Garg et al., 2021). For example, Ive et al. (2021) introduces a dynamic distribution over latent frequency classes as a reward signal. This distribution is shaped to promote more rare words in the policy search space. Static rewards are very often designed to assess generated text in terms of its fluency and adequacy. Fluency judgment assesses how a hypothesis satisfies the grammatical norms of a language. Adequacy judgment assesses how well a hypothesis conveys the meaning of the source sentence. The recent research performs both evaluations as semantic similarity assessments using the pre-trained contextualised embeddings such as BERT (Zhang et al., 2020; Mathur et al., 2019). For MT, semantic similarity assessment could be carried out using monolingual pre-trained embeddings against a reference, as in Gao et al. (2020), or using multilingual pre-trained embeddings in the unsupervised reference-less approach by considering the semantic similarity to source sentences (Wei et al., 2019; Song et al., 2021). We adopt the latter approach to measure adequacy.

3 Methodology

We start by formulating MT using RL, then introduce the Actor-Critic architectures and the reward functions used.

3.1 Neural Machine Translation (NMT)

A typical Neural Machine Translation (NMT) system is a Seq2Seq architecture (Sutskever et al., 2014; Bahdanau et al., 2014), where each source sentence ($\mathbf{X}$) is encoded by the encoder into a sequence of hidden states. At each decoding step t, a target word y_t is generated according to $p(y_t|\mathbf{y}_{<t}, \mathbf{X})$ conditioned on the input sequence $\mathbf{X}$ and decoded sequence $\mathbf{y}_{<t} = (y_1, \cdots, y_{t-1})$ up to the t-th time step:

$$\mathcal{L}_{mle} = \log p\left(y_t|\mathbf{y}_{<t}, \mathbf{X}\right) \quad (1)$$

3.2 Reinforcement Learning for NMT

In the RL framework, a Seq2Seq model is viewed as an agent and its parameters define the agent's **policy** (π). At each timestep (t), the agent observes the current **state** (s_t) of the **environment**, which is essentially the sequence of generated words from previous timesteps ($\hat{\mathbf{y}}_{1:t-1}$). Then, the agent's policy, which is based on the conditional probability $p\left(\hat{y}_t|\hat{\mathbf{y}}_{1:t-1}, \mathbf{X}\right)$, is used to select an **action** (a_t) at each timestep. In this context, an action is the candidate word ($\hat{y}_t$) chosen from the vocabulary. Subsequently, the environment adds the chosen word to the generated sequence, transitioning it to the next state ($\hat{\mathbf{y}}_{1:t}$). It also returns a reward (r_{t+1}) to the agent as an indication of how effective the chosen

word is. Hence, one possible training objective of the policy is to maximise the discounted sum of expected rewards from all timesteps:

$$\pi_* = \max_{\pi} \sum_{t=1}^{T} \gamma^{t-1} \mathop{\mathbb{E}}_{\hat{y}_t \sim \pi(\cdot|\hat{\mathbf{y}}_{1:t-1}, \mathbf{X})} [r_{t+1}] \quad (2)$$

where π_* denotes the optimal policy and γ is a constant discount factor. Under the policy π, one can formulate two functions: the *state value function* ($V_\pi(s_t)$) and the *state-action value function* ($Q_\pi(s_t, a_t)$). The former, $V_\pi(s_t)$, determines the effectiveness of the agent being in a particular state while the latter, $Q_\pi(s_t, a_t)$, indicates the effectiveness of selecting a certain action in that state:

$$V_\pi(s_t) = \mathop{\mathbb{E}}_{\hat{y}_t \sim \pi} [Q_\pi(s_t = \hat{\mathbf{y}}_{1:t-1}, a_t = \hat{y}_t)] \quad (3)$$

$$Q_\pi(s_t, a_t) = \mathop{\mathbb{E}}_{\pi} \left[\sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k} | s_t, a_t \right] \quad (4)$$

Hence the definition of the *advantage function* is:

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) \quad (5)$$

A training objective can aim to maximise the advantage function $\max_a A_\pi(s_t, a_t)$.

Alternatively, considering the definition of $A_\pi(s_t, a_t)$ in Equation (5), it implies that we can directly maximise the Q function:

$$\max_a A_\pi(s_t, a_t) \rightarrow \max_a Q_\pi(s_t, a_t) \quad (6)$$

The first objective (Equation (2)) has been used in REINFORCE-based methods (Sutton et al., 2000) such as the MIXER architecture (Ranzato et al., 2015). These methods sample trajectories, series of consecutive states, actions and rewards, and use their true returns to update the policy. As they use the true returns, they are considered to be unbiased. However, as an action in a certain state can be part of many trajectories with different returns, these methods are considered to have high variance (Sutton and Barto, 2014). To address this issue, Actor-Critic algorithms (Konda and Tsitsiklis, 2001) adopt the *Temporal Difference* (TD) learning method which performs *bootstrapping* by using only the immediate reward and estimated values to guide future action selection. The training objective used is Equation (6).

3.3 Actor-Critic Architectures

3.3.1 Actor-Critic with Q-Learning (ACQ) Model

An Actor-Critic model usually consists of an **actor** and a **critic** (Konda and Tsitsiklis, 2001). The two networks are neural networks parameterised by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. The actor acts as the policy of the model while the critic is a function approximation network. One simple variant (ACQ) of the AC architecture is trained by maximising the Q function. In this variant, the critic is defined as a Q network approximating the true Q function.

The actor's training objective is to maximise the probability of actions that yield high Q values. Using Q value estimates ($Q_{\boldsymbol{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}, w)$) computed by the main critic, the actor's policy loss ($\mathcal{L}_{policy}$) at each training timestep can be expressed as follows:

$$\mathcal{L}_{policy} = -[\frac{1}{N} \sum_{i=1}^{N} \sum_{t} \sum_{w \in \mathcal{W}} \pi_{\boldsymbol{\theta}}(w|\hat{\mathbf{y}}_{1:t-1,i}) Q_{\boldsymbol{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}, w)] \quad (7)$$

where N denotes the training batch size. The loss is calculated by summing over all the possible actions (w) in the entire vocabulary ($\mathcal{W}$). Following (Bahdanau et al., 2017), to avoid early policy determination and gradient vanishing issues, the final actor loss ($\mathcal{L}_{ACQ-actor}$) consists of the policy loss ($\mathcal{L}_{policy}$) and the Maximum Likelihood Estimation (MLE) loss ($\mathcal{L}_{mle}$) from cross-entropy training (XENT) (weighted by λ_{mle}). In other words, the addition of XENT is to address the problem of training collapse[1], commonly encountered when applying RL in language tasks.

$$\mathcal{L}_{ACQ-actor} = \mathcal{L}_{policy} + \lambda_{mle} \mathcal{L}_{mle} \quad (8)$$

The TD learning method, as mentioned previously, is used to train the critic network. It adopts the bootstrapping methodology which performs estimation based on other known estimates. The critic's training objective is to minimise the mean squared difference, called TD error, between all estimated Q values and their corresponding target values in each timestep. Intuitively, the critic is trained to be as good of a Q function approximator

[1] As pointed out by (Bahdanau et al., 2017), the MLE loss can help prevent early policy determination and vanishing gradient problems.

as possible:

$$\mathcal{L}_{TD} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t}(Q_{\phi}(\hat{\mathbf{y}}_{1:t-1,i}, \hat{y}_t) - \hat{Q}_{\bar{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}, \hat{y}_t))^2 \quad (9)$$

Each target $\hat{Q}_{\bar{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}, \hat{y}_t)$, expressed below, is defined as the sum of the immediate reward after generating $\hat{y}_t$ and the expected Q value of the proceeding timestep, which is computed using another Q network named the target critic:

$$\hat{Q}_{\bar{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}, \hat{y}_t) = r_{t+1} + \sum_{w\in\mathcal{W}} \pi_{\boldsymbol{\theta}}(w|\hat{\mathbf{y}}_{1:t,i}) Q_{\bar{\phi}}(\hat{\mathbf{y}}_{1:t,i}, w) \quad (10)$$

To ensure stability, the weights of the target critic ($\bar{\phi}$) are updated more slowly than the main critic with the linear interpolation between the current weights of the main and target critics. Also, following (Bahdanau et al., 2017), in addition to the TD error, the critic's loss ($\mathcal{L}_{ACQ-critic}$) contains an additional term, weighted by λ_{var}, which aims to minimise the variance in Q value estimation.

$$\mathcal{L}_{ACQ-critic} = \mathcal{L}_{TD} + \lambda_{var}\frac{1}{N}\sum_{i=1}^{N}\sum_{w\in\mathcal{W}} \left(Q_{\boldsymbol{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}, w) - \bar{Q}_{\boldsymbol{\phi}}(\hat{\mathbf{y}}_{1:t-1,i})\right)^2 \quad (11)$$

$$\bar{Q}_{\boldsymbol{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}) = \frac{1}{|\mathcal{W}|}\sum_{w'\in\mathcal{W}} Q_{\boldsymbol{\phi}}(\hat{\mathbf{y}}_{1:t-1,i}, w') \quad (12)$$

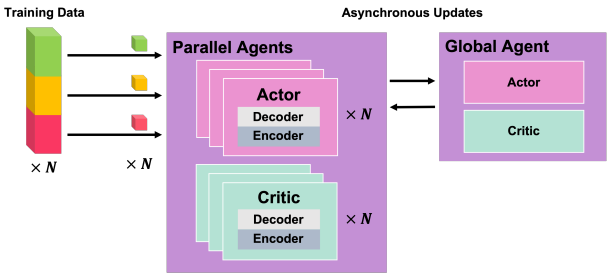

Figure 1: High-level structure of the Asynchronous Actor-Critic with Q-Learning Model (Async-ACQ) and the Asynchronous Advantage Actor-Critic (A3C) Model: multiple parallel agents and critics are trained independently. Their weights are used to update the weights of the global agent one by one.

Synchronous and Asynchronous ACQ Both the *asynchronous* and *synchronous* versions of the ACQ model can be easily constructed by deploying N actors with respective N critics on multiple threads. Each of the parallel actors is trained using different segments of the training dataset. For the synchronous variant, the weights of each of the actors are averaged to update the weights of the global agent. For the asynchronous variant, the global agent is updated by the local weights of each agent one by one. That is, during training, each thread-specific agent generates output sequences by sampling from its policy. Then, it performs loss computation and gradient accumulation until it reaches the pre-defined number of timesteps, in which it transfers the accumulated gradients to the global model. The global model subsequently performs an update on its parameters. As the last step of the asynchronous update, the parameters of the thread-specific agent invoking the update are synced with the parameters of the global model.

3.3.2 Advantage Actor-Critic (A2C) Model

Another variant of the AC model is the Advantage Actor-Critic (A2C) architecture (Konda and Tsitsiklis, 2001). In this model, the critic is defined as a function approximator, parameterised by $\boldsymbol{\psi}$, of the true V function. Compared to the first variant, the A2C model applies a different training objective to ACQ (Equation (6)).

Given the state space in language tasks is massive, calculating the expectation term would be computationally expensive or even impossible. Therefore, the advantage function can be approximated by sampling once.

$$A_{\pi}(s_t, a_t) \approx r_{t+1} + \gamma V_{\pi}(s_{t+1}) - V_{\pi}(s_t) \quad (13)$$

The actor network in the A2C model is trained in a similar fashion to that of ACQ. Here, the critic estimates the state values (i.e., $V_{\boldsymbol{\psi}}(\hat{\mathbf{y}}_{1:t-1,i})$ and $V_{\boldsymbol{\psi}}(\hat{\mathbf{y}}_{1:t,i})$) which are used by the actor to calculate the advantage value. The actor loss function ($\mathcal{L}_{A2C-actor}$) can be outlined as follows:

$$\mathcal{L}_{policy} = -[\frac{1}{N}\sum_{i=1}^{N}\sum_{t}\log \pi_{\boldsymbol{\theta}}(\hat{y}_t|\hat{\mathbf{y}}_{1:t-1,i}) A_{\boldsymbol{\psi}}(\hat{\mathbf{y}}_{1:t-1,i}, \hat{y}_t)] \quad (14)$$

$$A_{\boldsymbol{\psi}}(\hat{\mathbf{y}}_{1:t-1,i}, \hat{y}_t) = r_{t+1} + \gamma V_{\boldsymbol{\psi}}(\hat{\mathbf{y}}_{1:t,i}) - V_{\boldsymbol{\psi}}(\hat{\mathbf{y}}_{1:t-1,i}) \quad (15)$$

$$\mathcal{L}_{A2C-actor} = \mathcal{L}_{policy} + \lambda_{mle}\mathcal{L}_{mle} \quad (16)$$

Furthermore, compared to the first variant, the A2C critic is trained to minimise the TD error between its estimation and ground-truth data. The ground-truth data is essentially the true discounted reward-to-go (v_t).

$$\mathcal{L}_{A2C-critic} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t}\left(V_{\boldsymbol{\psi}}(\hat{\mathbf{y}}_{1:t-1,i}) - v_t\right)^2 \tag{17}$$

Synchronous and Asynchronous A2C The model setups for the synchronous and asynchronous A2C variants are analogous to the ACQ variants, in which N pairs of actors and critics are deployed.

In the **asynchronous A2C architecture (A3C)** (Mnih et al., 2016), the model also employs n-step TD Learning which uses the true returns from multiple steps in the advantage function to reduce the model bias (Sutton and Barto, 2014). The term n defines the number of steps to use the real rewards before bootstrapping (using the critic). The standard TD Learning would just use the immediate reward (1-step TD).

$$\begin{aligned} A_{\boldsymbol{\psi}'}(\hat{\mathbf{y}}_{1:t-1,i}, \hat{y}_t) = \sum_{\tau=0}^{n-1}\gamma^{\tau} r_{t+\tau} + \\ \gamma^n V_{\boldsymbol{\psi}'}(\hat{\mathbf{y}}_{1:t+n-1,i}) - V_{\boldsymbol{\psi}'}(\hat{\mathbf{y}}_{1:t-1,i}) \end{aligned} \tag{18}$$

where $\boldsymbol{\theta}'$ and $\boldsymbol{\psi}'$ represent the thread-specific parameters of each actor and critic, respectively.

4 Semantic-level Unsupervised Reward Function (SURF)

Our semantic-level unsupervised reward, SURF, is based on two scores: *Sentence Fluency* and *Sentence-level Semantic Similarity (SLSS)* (Song et al., 2021). Each score assesses translation quality of generated sequences from different aspects and is computed using a pre-trained model. To prevent reward sparsity, the reward function introduces a score normalisation mechanism which normalises scores of a generated sequence (from all timesteps) with respect to the score of its full target sequence. This subsequently yields an unsupervised reward function that is maximally dense. The Sentence Fluency score ($F(\hat{\mathbf{y}}_{1:t})$) is defined as the average log-likelihood of the generated sequence tokens ($\hat{\mathbf{y}}_{1:t}$) as defined by a pre-trained large LM.

The SLSS score measures the overall semantic similarity between the entire generated sequence and its source sequence calculated as the cosine similarity between the two sentence cross-lingual embeddings.

Score Normalisation From the RL perspective, the MT task does not define the environment component that the agent operates in. That is, unlike the classical RL setting where the environment is relatively fixed, the 'environment' in the MT task is mostly dependent on the source sequence, in terms of its sophistication, structure, length, etc. As a result, a valid and good translation of a source sequence would receive a relatively high score but is not directly comparable to other sequences due to the difference in source sentence complexity. Therefore, it is important to ensure that the reward function is uniform and generalised across all source sentences.

In order to do this for each source sequence, the reward function uses the corresponding target sequence as a 'soft' upper bound for what a machine-generated sequence could achieve. That is, for each of the two score metrics outlined above, the scores from all timesteps received by a generated sequence is normalised to the range 0 to 1 with respect to the score of the full target sequence. To demonstrate the normalisation method, let us consider the formulation below which uses the Sentence Fluency score metric as an example. Given a pair of source ($\mathbf{X}$) and target ($\mathbf{Y}$) sequences and a candidate sequence ($\hat{\mathbf{y}}_{1:t}$), the fluency scores from all timesteps of $\hat{\mathbf{y}}_{1:t}$ would be $\{F(\hat{\mathbf{y}}_1), F(\hat{\mathbf{y}}_{1:2}), ..., F(\hat{\mathbf{y}}_{1:t-1}), F(\hat{\mathbf{y}}_{1:t})\}$ while the fluency score for the entire reference target sequence ($\mathbf{Y}$) would be $F(\mathbf{Y})$. Using the fluency scores of the candidate and that of the reference, the normalised candidate scores can be calculated as follows:

$$F_{norm}(\hat{\mathbf{y}}_{1:i}) = \begin{cases} \frac{F(\hat{\mathbf{y}}_{1:i}) - min}{max - min} & \text{if} \quad max \neq min \\ 0.5 & \text{if} \quad max = min, \end{cases} \tag{19}$$

where

$$max = \max(\{F(\hat{\mathbf{y}}_1), ..., F(\hat{\mathbf{y}}_{1:t}), F(\mathbf{Y})\}) \tag{20}$$

$$min = \min(\{F(\hat{\mathbf{y}}_1), ..., F(\hat{\mathbf{y}}_{1:t}), F(\mathbf{Y})\}) \tag{21}$$

Considering the example formulation above, one can observe that the normalisation with respect to the reference score is considered as a 'soft' upper bound as it allows for candidate scores to be higher than the reference score (i.e., allowing the possi-

bility that candidate sequences can be better than their references).

The Pay-off Function and Reward Shaping Given a generated sequence $\hat{\mathbf{y}}_{1:t}$ at timestep t, its quality can be formulated as the Pay-off Function ($PO(\hat{\mathbf{y}}_{1:t}, \mathbf{X})$). The Pay-off Function, as expressed below, is based on the Sentence Fluency and SLSS scores described above.

$$PO(\hat{\mathbf{y}}_{1:t}, \mathbf{X}) = w_F \times e^{F(\hat{\mathbf{y}}_{1:t})} + w_S \times e^{SLSS(\hat{\mathbf{y}}_{1:t}, \mathbf{X})} \tag{22}$$

where, for simplicity, $F(\hat{\mathbf{y}}_{1:t})$ and $SLSS(\hat{\mathbf{y}}_{1:t}, \mathbf{X})$ denote the normalised Sentence Fluency and SLSS scores respectively. The terms w_F and w_S are fixed weights controlling the relative importance of Sentence Fluency and SLSS, respectively. It is important to emphasise that the exponential function is applied to each score since linearly adding each score leads to high variance and lower correlation with human scores (Song et al., 2021).

Using the Pay-off Function to determine the effectiveness of generating a token ($\hat{y}_t$) at timestep t, the final reward function is defined using reward shaping as the difference between the current Pay-off and the Pay-off of the previous timestep.

$$R(\hat{y}_t) = PO([\hat{\mathbf{y}}_{1:t-1}, \hat{y}_t], \mathbf{X}) - PO(\hat{\mathbf{y}}_{1:t-1}, \mathbf{X}) \tag{23}$$

5 Experimental Settings

5.1 Data

In our experiments, we used the German-English OpenSubtitles corpus (Lison and Tiedemann, 2016). There are approximately 14 million sequence pairs in this dataset extracted from subtitles of movies and TV shows, making it very diverse. The dataset was then divided into training, validation and test sets. The training set has approximately 13 million sentence pairs while each of the validation and test sets has roughly 5,000 sentence pairs.

The trained models were also tested on translating the IWSLT 2014 German-English test dataset, a popular dataset to benchmark RL-based methods. This dataset contains a parallel corpora with one reference per source sequence, obtained from TED talks (Cettolo et al., 2015). The test dataset contains approximately 6,000 pairs, with each sequence containing a few sentences of text. See Appendix A.2 for justification for treating the two datasets as coming from different domains.

5.2 Training

Following (Bahdanau et al., 2016), to ensure good initialisation of the model, the actor network is first pre-trained using XENT and the teacher forcing method. After that, the critic is pre-trained using TD Learning with the fixed pre-trained actor weights. At the last step, we train the actor and the critic jointly.

Generally, the actor architecture follows the OpenNMT Transformer architecture (Klein et al., 2017), with a few enhancements to enable step-wise decoding during training (i.e., action selection based on the model's previous outputs) and diversity in each agent's exploration strategy (see Appendix A.3 for more details). During training the actor selects a token at each timestep using the **Top-K** sampling method (Fan et al., 2018), in which a token is sampled from K tokens with the highest probabilities.

The critic architecture follows the Transformer architecture (Vaswani et al., 2017), with two major differences (see Appendix A.3.2 for detailed explanation). There are also two critic types, namely Q-critic and V-critic. The first critic type, Q-critic, is used in model variants with Q-Learning while the second type, V-critic, is used in other variants utilising the A2C architecture.

Multi-GPU Training When training synchronously in a multi-GPU environment, the model is deployed on a one-model-per-device basis to reduce training time. Each model has its own optimiser (we used Adam (Kingma and Ba, 2014)). During every update, the gradients are reduced and re-scaled across all devices to ensure that they are consistent across the models.

However, when training asynchronously, the global agent resides on one GPU device while three parallel agents are deployed on the remaining GPU devices. Instead of using one optimiser per agent, only a global optimiser is used. On every asynchronous update, the global optimiser updates the global model by using the gradients transferred from the parallel agent which invoked the update. The global optimiser used (SharedAdam) is a standard Adam optimiser modified to support multi-GPU communication.

Our formulation of the unsupervised reward uses the pre-trained OpenAI GPT Language Model from Hugging Face (Wolf et al., 2020). It also uses the Sentence Transformers tool (Reimers and Gurevych, 2020) with the XLM RoBERTa

model (Conneau et al., 2020) to generate the sentence embeddings. We do not expect the performance to change drastically by using other models.

During the joint actor-critic training, it took approximately one day to train each of the BLEU variants while the training time for SURF variants ranges from 3 to 5 days (see Appendix A.5 for the computational resource used). The increase in training time is mainly due to the usage of the large pre-trained models by SURF. We expect this time to reduce if a smaller language model is used. We leave this investigation to future work.

More details on the implementation and hyperparameters are given in Appendix A.3 and A.4.

Model configurations. We experimented using the nine configurations listed below:

- **Transformer** baseline (MLE, no RL) - state-of-the-art (SOTA) model;
- Synchronous ACQ RL architecture with the standard BLEU reward (**ACQ-BLEU**) and with our SURF (**ACQ**);
- Asynchronous ACQ RL architecture with the standard BLEU reward (**Async-ACQ-BLEU**) and with our SURF (**Async-ACQ**);
- Synchronous A2C RL architecture with the standard BLEU reward (**A2C-BLEU**) and with our SURF (**A2C**);
- Asynchronous A2C RL architecture with the standard BLEU reward (**A3Q-BLEU**) and with our SURF (**A3C**).

Each model was trained on the OpenSubtitles dataset and tested on both OpenSubtitles (in-domain) and IWSLT (out-of-domain) test sets.

By choosing this selection of models we are able to do the following: (a) generate the benchmark result using the Transformer baseline; (b) exhibit the advantage of SURF over the BLEU reward; and finally (c) explore the performance of SURF within the family of the multi-agent models (ACQ, Async-ACQ, A2C and A3C).

We used the standard set of MT evaluation metrics: BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014). We performed significance testing via bootstrap resampling using the `Multeval` tool (Clark et al., 2011).

6 Results

6.1 Performance on Automatic Evaluation Metrics

Results for the OpenSubtitles test set are in Table 1. They show that all of our model variants outperform the Transformer benchmark (+1.5 BLEU on average) with the A3C model performing the best (+2.5 BLEU vs. Transformer) while the Async-ACQ model is the second best performing variant being (+1.2 BLEU vs. Transformer). Mostly, our unsupervised reward contributes around +0.5 BLEU to the performance of each model when trained with the BLEU reward. Especially high improvement of +7.5 BLEU is observed for A3C. We attribute this result to the BLEU reward sparsity and hence impossibility to properly learn the relative improvement over the average for the actions as modelled by the advantage function. Note that Q-learning learns the absolute reward which is easier to model in this sparsity condition. This sparsity is particularly detrimental in the asynchronous setting, where the updates of parallel agents seem to exhibit too much variance to be useful. Those observations in the Advantage learning setting emphasise the important potential of our dense reward to elicit the benefits of different RL architectures.

Model	B	M	B	M
Transformer	33.3	29.9		
	BLEU		*SURF*	
ACQ	34.6*	30.7*	35.2*†	31.0*†
Async-ACQ	35.0*	30.9*	35.5*†	31.1*†
A2C	32.9*	29.9*	33.4*†	29.9*
A3C (Async-A2C)	28.3*	27.9*	**35.8***†	**31.2***†

Table 1: Results for the OpenSubtitles German-English test set. We report BLEU (B) and METEOR (M) scores. The symbol $\star$ indicates statically significant changes (p-value ≤ 0.05) as compared to the Transformer model while † indicates statically significant changes (p-value ≤ 0.05) as compared to the BLEU variant of the same RL-based architecture. The best result is highlighted in bold.

To probe the generalisation capacity of our models in the out-of-domain scenario, we have applied our models to the IWSLT test set. As shown in Table 2, the performance drop for the Transformer model is much higher than for our RL models (-6 BLEU vs. -2.4 BLEU on average). The Async-ACQ is the best-performing model on both metrics with the ACQ model being the second best (+6.3 BLEU and +5.8 BLEU vs. Transformer, re-

Model	B	M	B	M
MIXER‡ (Ranzato et al., 2015)	20.7	-	-	-
AC‡ (Bahdanau et al., 2016)	28.5	-	-	-
ERAC‡ (Dai et al., 2018)	29.0	30.6	-	-
SAC-BLEU‡ (Ive et al., 2021)	29.6	31.0	-	-
SAC-Unsuper‡ (Ive et al., 2021)	29.8	31.2	-	-
Transformer	27.3	29.5	-	-
	BLEU		*SURF*	
ACQ	32.4⋆	32.7⋆	33.1⋆†	33.1⋆†
Async-ACQ	32.8⋆	32.7⋆	**33.6**⋆†	**33.3**⋆†
A2C	29.4⋆	31.0⋆	30.2⋆†	31.0⋆
A3C (Async-A2C)	22.2⋆	27.6⋆	32.8⋆†	32.8⋆†

Table 2: Results for the IWSLT 2014 German-English test set. We report BLEU (B) and METEOR (M) scores. The symbol ⋆ indicates statically significant changes (p-value ≤ 0.05) as compared to the Transformer model while † indicates statically significant changes (p-value ≤ 0.05) as compared to the BLEU variant of the same RL-based architecture. The best result is highlighted in bold.

spectively). Mostly, our unsupervised reward contributes around +0.8 BLEU to the performance of each model. Especially high improvement of +10 BLEU is again observed for A3C showing the potential of SURF.

By way of offering a guideline of our model performance, we also report the scores of the previous SOTA on the IWSLT test set. Though those results are not directly comparable to our results as the pre-processing conditions are different: previous models have mainly applied a cut-off vocabulary implying the presence of unknown words in the training data while we are using the subword units that dispense us of the unknown words.

Note that, on both test sets, the asynchronous variants (Async-ACQ and A3C) performed better than their corresponding synchronous counterparts (ACQ and A2C respectively). We emphasise the potential of asynchronicity with our dense reward to positively influence performance.

Additionally, regarding the usage of Q- or V-critics: both Q-version Async-ACQ and V-version A3C have shown comparable performance on the OpenSubtitles dataset. However, the Q-version Async-ACQ has achieved better performance on the IWSLT test set. We hypothesise that this may be due to the fact that the Q-critic network in the ACQ architecture outputs the state-action values of the entire vocabulary at each timestep rather than a single state value (as in the V-critic network). Hence it performs a more fine-grained policy evaluation with lower variance in the critic outputs, leading to a more stable model overall. A more thorough investigation would lead to better insights.

6.2 Performance on Semantic-level Evaluation Metrics

As with the automatic evaluation results, similar conclusions could be drawn from the results of the assessment with the three semantic metrics used in the reward formulation: Sentence Fluency, Token-level Semantic Similarity (TLSS) and Sentence-level Semantic Similarity (SLSS) scores (See Appendix A.6 for the description of the TLSS score).

For the IWSLT test set, as shown on Table 4, there is a noticeable increase in the Fluency score across our models (in comparison to the Transformer). ACQ and Async-ACQ are also able to achieve distinctly better TLSS ans SLSS scores than the Transformer model. We observe that variants of ACQ and Async-ACQ models achieve similar performance. When comparing the BLEU and SURF variants, the BLEU variants of A2C and A3C models obtain higher Fluency scores but score less on TLSS and SLSS. This can be explained by fact that BLEU RL sentences are prone to be more verbose, repeating the same meaning in different words. The results for OpenSubtitles show similar tendencies (see Appendix A.7).

6.3 Human Evaluation

Finally, to gain deeper insights, we performed human evaluation on the translations of the OpenSubtitles and IWSLT 2014 test sets by the Transformer, Async-ACQ-BLEU and the best performing SURF variants (A3C for OpenSubtitles and Async-ACQ for IWSLT).

For this human analysis, we randomly selected 50 test samples from each test set. A rank of quality is assigned by the human evaluator (second author,

Source	all dies wurzelt in der mythologischen vergangenheit . das eigenartige an diesen großen häusern , in denen aufgrund der mischehen sechs oder sieben sprachen gesprochen werden , ist jedoch , dass man niemals jemanden hört , der eine sprache lernt .
Target	and this is all rooted in the mythological past , yet the curious thing is in these long houses , where there are six or seven languages spoken because of intermarriage , you never hear anyone practicing a language .
Transformer	all this mythology in the mythological context , which is curious about these great houses , judging by the patter of six or seven languages , however , you never hear anyone learning a language .
Async-ACQ-BLEU	the strange thing about these big houses where they speak six or seven languages , is that you never hear anyone who learns a language .
Async-ACQ	*all this rambling in the mythological past , the curious thing about these big houses where they speak six or seven languages based on the basic language is that you never hear anyone who learns a language.*

Table 3: Examples of translation generated by Transformer, ACQ-BLEU and Async-ACQ. We also report the original source sequence (SOURCE) and its reference (TARGET). The best translation is highlighted in italics.

Model	Fluency	TLSS	SLSS	Fluency	TLSS	SLSS
Transformer	1.024	2.454	2.339	-	-	-
		BLEU			*SURF*	
ACQ	1.029*	2.456*	2.350*	**1.029***†	**2.457***†	**2.357***†
Async-ACQ	1.032*	2.455*	2.350*	1.029*†	2.456*†	2.350*†
A2C	1.035*	2.451*	2.300*	1.027*†	2.454*†	2.339*†
A3C (Async-A2C)	1.047*	2.422*	2.267*	1.027*†	2.454*†	2.342*†

Table 4: Results for the IWSLT 2014 German-English test set. We report Sentence Fluency, Token-level Semantic Similarity and Sentence-level Semantic Similarity scores. Also, the symbol $\star$ indicates statically significant changes (p-value ≤ 0.05) as compared to the scores of Transformer model while † indicates statically significant changes (p-value ≤ 0.05) as compared to the BLEU variant of the same RL-based architecture. The best result is highlighted in bold. Note that some of the improvements are beyond the displayed precision of 3 decimal points.

Test Set	Transformer	BLEU	SURF
OS	0.10	0.08	**0.20**
IW	0.06	0.48	**0.60**

Table 5: Human ranking results comparing the OpenSubtitles (OS) test outputs for Async-ACQ-BLEU and A3C and the IWSLT 2014 (IW) test outputs for Async-ACQ-BLEU and Async-ACQ. The best result is highlighted in bold.

fluent speaker of both English and German) from 1 to 3, allowing ties. Following the common practice in MT, each system was then assigned a score which reflects how often on average it was judged to be better or equal to other systems (Bojar et al., 2017). Table 5 shows that most of our variants have higher evaluation scores than the Transformer model. In particular, on the IWSLT test set, both Async-ACQ variants outperform the Transformer by a large margin. As compared to the best BLEU model, the A3C and Async-ACQ models perform significantly better on both the OpenSubtitles and the IWSLT test sets (+0.12 point). Table 3 shows translations generated by the three models on the IWSLT test set. Note that Async-ACQ demonstrates the best fluency and adequacy.

7 Conclusion

We have presented SURF, a new unsupervised semantic-level reward function, efficiently addressing the reward sparsity issue and mimicking human evaluation by considering both sentence fluency and semantic similarity. We have explored this reward for a new family of Actor-Critic Transformer-based Architectures with synchronous and asynchronous variants that promote the exploration of the search space. We demonstrate that SURF shows strong potential to elicit the benefits of various RL architectures. Our results show that it outperforms the traditional sparse BLEU reward for the same architectures in the automatic, semantic-level and human evaluation. Our code is available at https://github.com/AtomAnu/SURF.

There are several directions to take our work further: we can investigate the utility of our reward for other architectures and we can also explore different sampling strategies for each of the agents of our multi-agent models. Finally, we have investigated only two datasets to ensure comparability to the existing benchmarks. A more thorough investigation with more datasets and language pairs is needed to fully assess the scope of our contribution.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. Cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 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.

M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and Marcello Federico. 2015. Report on the 11 th iwslt evaluation campaign , iwslt 2014.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. 2020. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Zihang Dai, Qizhe Xie, and Eduard Hovy. 2018. From credit assignment to entropy regularization: Two new algorithms for neural sequence prediction. *arXiv preprint arXiv:1804.10974*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. pages 889–898.

Yang Gao, Wei Zhao, and Steffen Eger. 2020. SUPERT: Towards new frontiers in unsupervised evaluation metrics for multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1347–1354, Online. Association for Computational Linguistics.

Sonal Garg, Sumanth Prabhu, Hemant Misra, and G. Srinivasaraghavan. 2021. Unsupervised contextual paraphrase generation using lexical control and reinforcement learning.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4099–4106. International Joint Conferences on Artificial Intelligence Organization.

Julia Ive, Zixu Wang, Marina Fomicheva, and Lucia Specia. 2021. Exploring supervised and unsupervised rewards in machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1908–1920, Online. Association for Computational Linguistics.

Samuel Kiegeland and Julia Kreutzer. 2021. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1673–1681, Online. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Vijay Konda and John Tsitsiklis. 2001. Actor-critic algorithms. *Society for Industrial and Applied Mathematics*, 42.

Pierre Lison and Jörg Tiedemann. 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).

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2019. Putting evaluation in context: Contextual embeddings improve machine translation evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2799–2808, Florence, Italy. Association for Computational Linguistics.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

Yurun Song, Junchen Zhao, and Lucia Specia. 2021. SentSim: Crosslingual semantic evaluation of machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3143–3156, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

R. S. Sutton and A. G. Barto. 2014. *Reinforcement Learning: An Introduction*, second edition edition. Cambridge, Massachusetts: The MIT Press.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 2–6. Curran Associates, Inc.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Shuo Wang, Zhaopeng Tu, Zhixing Tan, Shuming Shi, Maosong Sun, and Yang Liu. 2021. On the language coverage bias for neural machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4778–4790, Online. Association for Computational Linguistics.

Xiangpeng Wei, Yue Hu, Luxi Xing, and Li Gao. 2019. Unsupervised neural machine translation with future rewarding. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 281–290, Hong Kong, China. Association for Computational Linguistics.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Appendix

A.1 Ethics Considerations

We are aware of the discussions around the risks related to unintended harmful effects and uses, environmental consequences, fairness and privacy considerations of large language models in general (Bender et al., 2021), and machine translation models specifically (Wang et al., 2021). We note here that our models constitute a primarily theoretical contribution and were trained and tested on standard datasets. Before deployment in a production setting our methodology is subject to retraining with data pre-processed in the appropriate way (as our model is not equipped with relevant security, privacy and fairness mechanisms), systematic debugging, extensive simulation, testing and validation under the supervision of experts.

A.2 Domain Distance

To justify that the IWSLT test set is indeed considered out-of-domain, we have trained a German language model using the source sentences (in German) from the OpenSubtitles training set. For this we used the fairseq toolkit (Ott et al., 2019). The resulting difference in language model perplexity values for the OpenSubtitles and IWSLT test sets (45.52 and 555.71, respectively) is important enough to justify that IWSLT is considered out-of-domain.

A.3 Implementation Details

A.3.1 The Actor

OpenNMT Transformer Implementation In the OpenNMT framework (Klein et al., 2017), the Transformer architecture is implemented slightly differently from the original architecture in (Vaswani et al., 2017). Its implementation follows the up-to-date implementation of the tensor2tensor framework (Vaswani et al., 2018), created by the authors of (Vaswani et al., 2017). The main difference lies in the normalisation technique used in the Transformer. That is, pre-normalisation is applied in each sub-layer of the encoder and the decoder instead of post-normalisation. The output of each sub-layer with pre-normalisation can be expressed as follows:

$$x + Sublayer(LayerNorm(x)) \tag{24}$$

In the original architecture where post-normalisation is used, layer normalisation ($LayerNorm$) is applied after the summation ($x + Sublayer(x)$), as shown below:

$$LayerNorm(x + Sublayer(x)) \tag{25}$$

Step-wise Decoding and Exploration Strategies During the joint AC training, instead of just computing the policy distributions as the output, the actor would perform step-wise decoding by selecting a token to generate at each timestep, given the encoded source sequence and the previously generated tokens. As mentioned before, this is done to ensure that there would be no exposure bias during inference as the model is trained to condition the generation process using its own outputs.

To allow each agent to be diverse in their exploration strategies, the actor can operate in two possible main modes of token selection. In the first mode, the actor selects a token at each timestep using the **Top-K** sampling method (Fan et al., 2018), in which a token is sampled from K tokens with the highest probabilities. In the second mode of operation, the actor performs token selection based on **Nucleus** sampling (Holtzman et al., 2020). In the Nucleus sampling method, a token is chosen from the smallest possible set of tokens that has an accumulated probability equal or higher than a pre-defined probability value (p). For instance, if p is set to 1, the actor would perform token selection from the entire vocabulary. Similarly, if the value of K for Top-K sampling is set to the vocabulary size, the actor would sample from the entire vocabulary as well.

The actor also incorporates the notion of **Temperature** to further increase the diversity of exploration strategies. A pre-defined value of Temperature ($temp$) is used to increase the probability of

probable tokens while reducing others that are not probable.

$$p(w_i|\hat{\mathbf{y}}_{1:t-1}) = \frac{\exp\left(l_i/temp\right)}{\sum_j \exp\left(l_j/temp\right)} \quad (26)$$

where l_i is the logit of the token w_i. One can observe that, as $temp$ decreases, the probability of probable tokens would increase. If $temp$ is set to 1, the above expression would simplify to the normal softmax operation.

We have empirically found that the sampling Temperature should not be applied in conjunction with Top-K or Nucleus sampling as it leads to highly greedy policies, especially when K or p is already low. It should be applied on when the agent samples from the entire vocabulary. After experimenting with different configurations of exploration strategies, we found that Top-K sampling was the most effective.

A.3.2 The Critic

The critic architecture follows the Transformer architecture, with two major differences.

Reference Encoding The first difference between the actor and the critic is that the encoder of the critic encodes the reference sequences instead of the source sequences. The reason is to allow to critic to evaluate each generated sequence by comparing with its reference sequence.

Output Layer The second difference is the output layer used in the critic. In the Q-critic model, its output layer is a one-layer feed-forward network with the output dimension equal to the vocabulary size of the target language. This is because the Q-critic model outputs the state-action value (i.e., Q-value) for every word in the vocabulary.

For the V-critic model, its output layer also contains a one-layer feed-forward network with the output size of 1. Given a generated sequence, the critic outputs the state value (i.e., V-value) for each token in that sequence.

A.4 Hyper-parameters

A.4.1 Actor Pre-training

Table 6 lists all the hyper-parameters used during actor pre-training.

A.4.2 Critic Pre-training

Table 7 lists all the hyper-parameters used during critic pre-training.

Hyper-parameter	Value
Optimizer	Adam
Adam Beta 1	0.9
Adam Beta 2	0.998
Learning Rate	2
LR Decay Method	noam
Warmup Steps	6000
Batch Size	4096
Gradient Accumulation Steps	3
Source Vocabulary Size	100000
Target Vocabulary Size	100000
Word Embedding Size	512
Hidden Layers Size	512
Encoder Layers	6
Decoder Layers	6
Attention Heads	8

Table 6: List of the hyper-parameters used during the actor pre-training stage.

Hyper-parameter	Value
Optimizer	Adam
Adam Beta 1	0.9
Adam Beta 2	0.998
Learning Rate	0.001
LR Decay Rate	0.9
LR Decay Steps	1000
Batch Size	4096
Gradient Accumulation Steps	3
Source Vocabulary Size	100000
Target Vocabulary Size	100000
Word Embedding Size	512
Hidden Layers Size	512
Encoder Layers	6
Decoder Layers	6
Attention Heads	8
γ (Discount Factor)	0.99
λ_{var} (Q-critic)	0.25
Multi-step Return (V-critic)	5
w_F (Sentence Fluency Weight)	1
w_S (SLSS Weight)	1

Table 7: List of the hyper-parameters used during the critic pre-training stage.

A.4.3 Joint Actor-Critic Training

Table 8 lists all the hyper-parameters used during synchronous and asynchronous joint Actor-Critic training.

A.5 Computational Resource

Each of our models was trained on a GPU-accelerated instance with four NVIDIA V-100 SXM2 GPUs.

A.6 Token-level Semantic Similarity

The Token-level Semantic Similarity (TLSS) score is used as one of the semantic-level evaluation metrics. It can be used as an additional score metric in the reward function as well. TLSS measures the semantic similarity between tokens in the gener-

Hyper-parameter	Value	
Training Mode	Sync	Async
Optimizer	Adam	SharedAdam
Adam Beta 1	0.9	0.9
Adam Beta 2	0.998	0.998
Actor LR	0.000001	0.000001
Critic LR	0.00001	0.00001
LR Decay Rate	0.9	0.9
LR Decay Steps	1000	1000
Batch Size	2000	2000
Grad. Accum.	1	1
γ	0.99	0.99
λ_{xent}	0.01	0.01
λ_{var}	0.25	0.25
Multi-step Return	5	5
w_F	1	1
w_S	1	1
K	300	[100,200,500]
ϵ	0.2	-

Table 8: List of listing the hyper-parameters used during synchronous and asynchronous joint Actor-Critic training stage.

ated sequence and its source sequence. Following the methodology adopted in the SentSim evaluation metric (Song et al., 2021), each token in the source and generated sequences is passed to a cross-lingual language model to obtain its cross-lingual embedding. Then, each token in the source sequence (x_i) is matched to a token in the generated sequence ($\hat{y}_j$) with the highest cosine similarity value to compute the recall. Similarly, the precision value is computed by matching each token in the generated sequence to a token in the source sequence based on cosine similarity. As a results, the recall and precision of a generated sequence can be expressed as follows:

$$R(\hat{\mathbf{y}}_{1:t}, \mathbf{X}) = \frac{1}{|\mathbf{X}|} \sum_{x_i \in \mathbf{X}} \max_{\hat{y}_j \in \hat{\mathbf{y}}_{1:t}} \mathbf{x}_i^{nxe} \cdot \hat{\mathbf{y}}_j^{nxe} \quad (27)$$

$$P(\hat{\mathbf{y}}_{1:t}, \mathbf{X}) = \frac{1}{|\hat{\mathbf{y}}_{1:t}|} \sum_{\hat{y}_j \in \hat{\mathbf{y}}_{1:t}} \max_{x_i \in \mathbf{X}} \mathbf{x}_i^{nxe} \cdot \hat{\mathbf{y}}_j^{nxe} \quad (28)$$

where $\mathbf{x}_i^{nxe}$ and $\hat{\mathbf{y}}_j^{nxe}$ denote the pre-normalised (i.e., $\mathbf{x}_i^{nxe} = \frac{\mathbf{x}_i^{xe}}{||\mathbf{x}_i^{xe}||}$) cross-lingual embeddings of x_i and $\hat{y}_j$, respectively. Using the precision and recall, the final TLSS score is defined as the F1 measure.

$$TLSS(\hat{\mathbf{y}}_{1:t}, \mathbf{X}) = F(\hat{\mathbf{y}}_{1:t}, \mathbf{X}) \quad (29)$$

$$= 2\frac{P(\hat{\mathbf{y}}_{1:t}, \mathbf{X}) \cdot R(\hat{\mathbf{y}}_{1:t}, \mathbf{X})}{P(\hat{\mathbf{y}}_{1:t}, \mathbf{X}) + R(\hat{\mathbf{y}}_{1:t}, \mathbf{X})} \quad (30)$$

The TLSS scores are computed using the BERTScore tool (Zhang et al., 2020).

A.7 Semantic-level Evaluation on the OpenSubtitles Test Set

For the OpenSubtitles test set, as shown in Table 9, there is a slight increase in the Fluency scores for all our model as compared to the Transformer. There are more apparent increases in the TLSS and SLSS scores. Among all the variants, the ACQ model achieves the highest on all three scores. The Async-ACQ model is the second best with its scores being very close to the scores of the ACQ model. When comparing the reward functions, the BLEU and SURF variants of the ACQ and Async-ACQ models achieve similar performance. However, for the A2C and A3C models, the BLEU variants achieve higher Fluency scores than the SURF variants but their SLSS scores are noticeably lower than that of SURF. This can be explained by fact that BLEU RL sentences are prone to be more verbose, repeating the same content. This was observed during human evaluation (See Subsection 6.3).

Model	Fluency	TLSS	SLSS	Fluency	TLSS	SLSS
Transformer	1.036	2.450	2.360	-	-	-
		BLEU			*SURF*	
ACQ	1.037*	2.455*	2.384*	**1.037***†	**2.456***†	**2.386***†
Async-ACQ	1.037*	2.455*	2.387*	1.037*†	2.455*†	2.385*†
A2C	1.039*	2.455*	2.377*	1.037*†	2.454*†	2.379*†
A3C (Async-A2C)	1.043*	2.443*	2.347*	1.037*†	2.454*†	2.383*†

Table 9: Results for the OpenSubtitles German-English test set. We report Sentence Fluency, Token-level Semantic Similarity and Sentence-level Semantic Similarity scores. Also, the symbol $\star$ indicates statically significant changes (p-value ≤ 0.05) as compared to the scores of Transformer model while † indicates statically significant changes (p-value ≤ 0.05) as compared to the BLEU variant of the same RL-based architecture. The best result is highlighted in bold. Note that some of the improvements are beyond the displayed precision of 3 decimal points.