Master's Thesis

# SENTENCE LEVEL ACTOR-CRITIC METHOD VIETNAMESE-ENGLISH NEURAL MACHINE TRANSLATION.

1910443 Nguyen Viet Phuong

Supervisor Prof. Nguyen Minh Le

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

October 2022

# Abstract

When a piece of text is automatically translated from one language to another, this process is known as machine translation (MT). Neural machine translation refers to a solution that uses neural networks to translate text (NMT). The machine translation dataset includes not just one but two languages: the source language and the destination language. This is in contrast to other language models, where the corpus contains only one language. In short, connections between each sentence in the original language and its translated counterpart in the target language are established, and after that, these connections are used to predict translated sentences from the source sentence.

Most optimization algorithms for NMT will use token-level maximum likelihood estimation during neural machine translation training to optimize the model. However, during evaluation sequence generation, like beam search, use only the probability distribution for each time step to infer the translated sentence. The translated sentence is subsequently assessed by a corpus-level held-out set evaluator using metrics such as the BLEU score, which cannot be differentiated or decomposed.

The use of reinforcement learning in neural machine translation is expected to lessen the discrepancy between training and evaluation. However, the model is still limited by sparse rewards, which will affect the model's quality. The actor-critic method will be used to enrich the rewards when training reinforcement for neural machine translation. To achieve this goal, we propose an actor-critic approach to the sentence-level machine translation model using the BLEU score as the goal to improve the translated sentences. In this article, we achieve remarkable progress on the translation task Vietnamese-English and vice versa using PhoMT and IWSLT 2015 data sets.

**Keywords:** Neural Machine Translation, Reinforcement Learning, Actor-critic method.

# Acknowledgment

First and foremost, I would like to thank my major supervisor, Professor NGUYEN LE MINH, for not only providing me with the opportunity to study at JAIST but also for creating the most favorable conditions for me to accomplish this thesis. Lastly, I'd like to thank everyone at NGUYEN-laboratory sensei's for everything they've done to assist me succeed here.

# List of Figures

# List of Tables

# Contents

This thesis was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and Vietnam National University Ho Chi Minh City - University of Science.

# Chapter 1

# Introduction

Today, the need to exchange information between countries and cultures is increasing, making the need for translation a necessity. The manual translation process by humans gives high quality but slow speed, low productivity, and high cost, which cannot be reused. Moreover, an interpreter, no matter how good, cannot translate well in all fields and different languages. Therefore, an automatic computer translation system is needed to help with the process of translation.

## 1.1 Problem statement

These days, more and more people are using neural machine translation (NMT) due to its excellent performance and the fact that no manual technical work is required. Most models will use supervised training to train, which means by using the source sentence and the prior (ground-truth) target tokens as inputs, it is typically trained to increase the likelihood of each token in the target phrase. Such training approach is referred as maximum likelihood estimation (MLE) [1].The token-level goal function during training is simple to use but is incompatible with sequence-level evaluation metrics like BLEU [2].
The improvement of sequence-level objectives has been implemented using reinforcement learning (RL) techniques to overcome the consistency problem. For instance, for NMT sequence generation tasks, policy optimization techniques like REINFORCE and actor-critic are used [3].

Recently paper by Kiegeland et al. [4] claims "Training from scratch will fail any method that uses the BLEU score as the reward signal because there will be no non-zero reward translation outputs to sample. The practical advantages over a strong pre-trained model vanish when there is little to learn from the new feedback, as when it is provided on the same data that the model was already trained on. Reinforcement learning methods have

the potential to enhance machine translation models in settings when no reference translations are available, only reward signals are, and models can be pre-trained using already collected data." So in this study we will define reward signal that can be train and environment of reinforcement learning apply to neural machine translation.

## 1.2 Objectives

We will design an algorithm that makes pre-trained neural machine translation possible to further improve the BLEU score by $\Delta$ points, higher is better.

## 1.3 Purpose of this study

Apply Actor-Critic method ( One kind of Reinforcement Learning ) to Neural Machine Translation where Actor is neural machine translation model such as Transformer. Critic is model from paper by Lee et al [5]. Structure of this method similar to method of paper by Yang et al. [6] but instead of label 0 and 1 for training discriminator we use BLEU score for train critic (discriminator) model. And this study is specialized for Vietnamese-English datasets.

## 1.4 Thesis Outline

1. **Chapter 2** We introduce the data used for this study and the basics of how machine translation models to train and inference, as well as some knowledge about reinforcement learning.
2. **Chapter 3** We detail how the actor-critic approach was applied to the neural machine translation model in our experiments.
3. **Chapter 4** We talk about experimental setup and experimental results.
4. **Chapter 5** We talk about the weaknesses of the current method and new ideas for future work that need improvement.

# Chapter 2

# Background and Related Datasets

## 2.1 Related Datasets

The data set is the most crucial part of a machine learning system. With a quality data set, the machine learning model may effectively learn the critical information present in the data. The following is an introduction to some of the most well-known datasets for neural machine translation for English-Vietnamese and vice versa.

### 2.1.1 IWSLT 2015 dataset

In 2015, Stanford developed the International Words and Sentences Translation (IWSLT) corpus, which consists of many pairs of languages, one of them being a bilingual dataset in English and Vietnamese. Machine translation models for the present English-to-Vietnamese and vice versa challenge are compared and evaluated using this dataset https://paperswithcode.com/sota/machine-translation-on-iwslt2015-english-1Machine Translation Task on IWSLT'2015 English-Vietnamese.

| Data set | Sentences | Download |
|---|---|---|
| Training | 133,317 | Github link |
| Development | 1,553 | Github link |
| Test | 1,268 | Github link |

Table 2.1: IWLST 2015 En-Vi dataset

### 2.1.2 PhoMT dataset

Because in deep learning, no matter how good the algorithm is, the most important part is still the data. Before 2021, there were not many bilingual datasets for Vietnamese-English open source, only IWSLT 2015. Also, for

the same reason. That is why machine translation for Vietnamese-English is classified as low-resource, making it difficult for researchers. So in 2021VinAI created PhoMT open-source contains 3.02M parallel sentence pairings in Vietnamese and English and is a high-quality, large-scale parallel dataset. The following are some statistics from the dataset: 2.1:

| Domain | Total | | Training | | | Validation | | | Test | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | #doc | #pair | #pair | #en/s | #vi/s | #pair | #en/s | #vi/s | #pair | #en/s | #vi/s |
| News | 2559 | 41504 | 40990 | 24.4 | 32.0 | 257 | 22.3 | 30.3 | 257 | 26.8 | 34.5 |
| Blogspot | 1071 | 93956 | 92545 | 25.0 | 34.6 | 597 | 26.4 | 37.8 | 814 | 23.7 | 31.5 |
| TED-Talks | 3123 | 320802 | 316808 | 19.8 | 23.8 | 1994 | 20.0 | 24.6 | 2000 | 22.0 | 27.9 |
| MediaWiki | 38969 | 496799 | 490505 | 26.0 | 32.8 | 3024 | 25.3 | 32.3 | 3270 | 27.0 | 33.7 |
| WikiHow | 6616 | 513837 | 507379 | 18.9 | 22.4 | 3212 | 17.9 | 21.5 | 3246 | 17.5 | 21.5 |
| OpenSub | 3312 | 1548971 | 1529772 | 9.7 | 11.1 | 9635 | 9.5 | 10.7 | 9564 | 10.0 | 11.4 |
| All | 55650 | 3015869 | 2977999 | 15.7 | 19.0 | 18719 | 15.3 | 18.7 | 19151 | 16.2 | 19.8 |

Figure 2.1: Dataset statistics of PhoMT.

| Data set | Sentences |
|---|---|
| Training | 2,977,999 |
| Development | 18,719 |
| Test | 19,151 |

Table 2.2: PhoMT En-Vi dataset

## 2.2 Background Knowledge

### 2.2.1 Tokenization: How text is represented

In the context of natural language processing tasks, the term "tokenization" refers to the process by which a string of text, such as "A set of words which creates a complete meaning is termed a sentence" is represented as a sequence of vocabulary components (called tokens).

**Character-level tokenization.** Let $V$ stand for the **alphabet**, for example (plus punctuation). A sequence of length 69 would result from the preceding example: ['A', ' ', 's', 'e', ...]. Tokenization at the character level typically produces exceedingly long sequences.

**Word-level tokenization.** Similarly, we could have $V$ be made up of nothing but English words (plus punctuation). A sequence of length 13

would result from the preceding example: [‘A ’, ‘set ’, ‘of ’, ...]. Tokenization at the word level usually necessitates a colossal vocabulary and has trouble with test-time additions of new words.

**Subword tokenization.** This is the current standard practice: $V$ is a collection of frequently occurring word segments such as "ful," "ing," and "pre." Common words like "are" are generally tokenized separately, and $V$ also includes single characters to guarantee that every possible word is represented. Subword tokenization can be done in various ways. One of the simplest and most successful ones is Byte Pair Encoding [8]. There are two implement of BPE tool first is by author https://github.com/rsennrich/subword-nmt.git, second is tool by Google called SentencePiece https://github.com/google/sentencepiece.git.

**Final vocabulary and text representation.** Each tokenization/vocabulary element is given a distinct index in the range $\{1, 2, \ldots, N_{\mathrm{V}} - 3\}$. The vocabulary is then expanded to include a variety of specialized tokens. A couple of the possible unique tokens are discussed here: Beginning and ending sequence tokens are represented by `bos_token` $:= N_{\mathrm{V}} - 1$ , and `eos_token` $:= N_{\mathrm{V}}$ , $N_{\mathrm{V}} = |V|$ represents the number of words in the full lexicon.

To represent text, we use a pair of tokens, the `bos_token` and the `eos_token`, which are followed by a series of indexes (called emphtoken IDs) that correspond to the text's (sub)words.

**Token embedding.** The token embedding is responsible for learning how to properly represent each vocabulary element as a vector in $\mathbb{R}^{d_{\mathrm{e}}}$ see 1.

**Algorithm 1:** How to embed token.

**Input:** $v \in V \cong [N_{\mathrm{V}}]$, ID of token.
**Output:** $\boldsymbol{e} \in \mathbb{R}^{d_{\mathrm{e}}}$,the token's scalar representation in vector form.
**Parameters:** $\boldsymbol{W_e} \in \mathbb{R}^{d_{\mathrm{e}} \times N_{\mathrm{V}}}$, Matrix for tokens in vocab.
1 **return** $\boldsymbol{e} = \boldsymbol{W_e}[:, v]$

**Unembedding.** Distribution over the vocabulary elements is learned by the unembedding from a vector representation of a token and its context; see 2.

**Algorithm 2:** Unembedding.

**Input:** $\boldsymbol{e} \in \mathbb{R}^{d_e}$, a token encoding.
**Output:** $\boldsymbol{p} \in \Delta(V)$, a statistical distribution of words.
**Parameters:** $\boldsymbol{W_u} \in \mathbb{R}^{N_V \times d_e}$, the unembedding matrix.
1 **return** $\boldsymbol{p} = \text{Softmax}(\boldsymbol{W_u e})$

### 2.2.2 Encoder-decoder framework seq2seq model

**Encoder.** In order to compress all of the information from the input string into a vector of a specified length (the "intermediate vector"), the encoder goes over each token in the string individually. This vector is then transformed by the encoder and sent on to the decoder.

**Context Vector.** This vector helps the decoder arrive at the correct conclusion by encapsulating the complete meaning of the input string and providing it with context. This is the first hidden state of the decoder, which also serves as the latent, hidden state of the sequence. It is computed by the encoder, which also serves as the latent, hidden state of the sequence.

**Decoder.** Uses the context vector and tries to predict the target sequence.

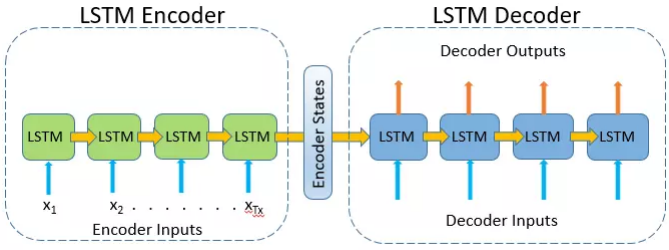


Figure 2.2: Encoder decoder framework example by LSTM.

### 2.2.3 Transformer

In the past, people used neural network architectures such as convolution neural network (CNN) and feedback neural network (RNN) applied to seq2seq. The advantages and disadvantages of these two network architectures can be summarized as follows:

- Even though CNN networks can be built in parallel at a single layer, they cannot capture sequence dependencies of varying lengths.
- RNNs can learn to recognize information that is spread out throughout a sequence of varying length, but they are unable to parallelize a sequence.

An attempt to pool the benefits of convolutional neural networks (CNNs) and recurrent neural network (RNN), Vaswani *et al.* [9] designed a new architecture using attention mechanism. This architecture, called Transformer 2.3, parallelizes by learning the feedback sequence with a attention mechanism, and also encodes the position of each element in the sequence. Now we have a model that works with less iterations in training.

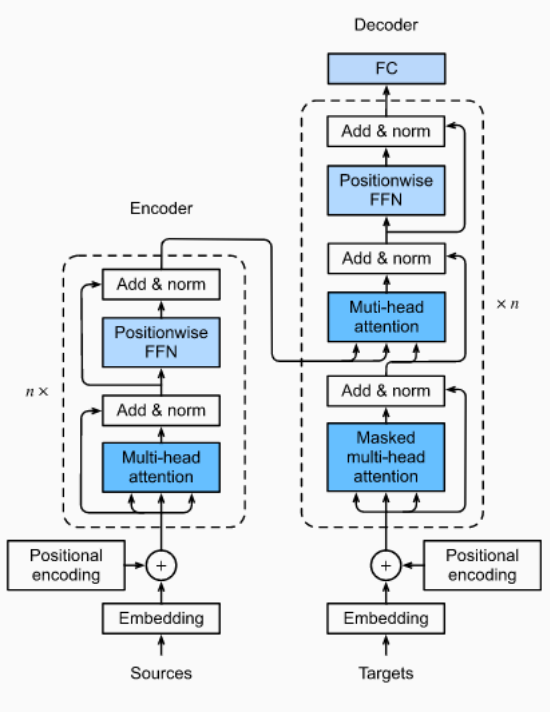


Figure 2.3: The transformer architecture.

### 2.2.4 Training and Inference in NMT

**Training.** By using the source sentence and the previous (ground-truth) target tokens as inputs, it is typically trained to maximize the likelihood of each token appearing in the target sentence. This is accomplished through the training process. Maximum likelihood estimate is the name given to this

type of training approach (MLE) [1]. For training seq2seq model we need to maximize below function 2.1:

$$\max J(\theta) := \sum_{(x,y)} \log P(y \mid x; \theta) = \sum_{(x,y)} \sum_{t=1}^{T_y} \log P\left(y_t \mid y_{<t}, x; \theta\right) \tag{2.1}$$

**Inference.** After having the probability distribution, we will use it to predict the translated sentence from the input of the sentence to be translated, see 2.2 and 3 for more detail. Current $\hat{y}_t$ depends on the previous generated $\hat{y}_{<t}$.

$$\hat{y}_t = \operatorname{argmax} P\left(\cdot \mid \hat{y}_{<t}, x; \theta\right) \tag{2.2}$$

A

**Algorithm 3:** $\hat{\boldsymbol{x}} \leftarrow$ `Inference`$(\boldsymbol{z}, \hat{\boldsymbol{\theta}})$

`/* Performing the prediction with a trained sequence-to-sequence model. */`

**Input:** A seq2seq transformer and trained parameters $\hat{\boldsymbol{\theta}}$ of the transformer.

**Input:** $\boldsymbol{z} \in V^*$, input sequence, e.g. a sentence in English.

**Output:** $\hat{\boldsymbol{x}} \in V^*$, output sequence, e.g. the sentence in Vietnamese.

**Hyperparameters:** $\tau \in (0, \infty)$

$\hat{\boldsymbol{x}} \leftarrow$ [`begin of sentence token`]
$y \leftarrow 0$
**while** $y \neq$ *end of sentence* **do**
  $\boldsymbol{ProbDis} \leftarrow$ `Transformer`$(\boldsymbol{z}, \hat{\boldsymbol{x}} \mid \hat{\boldsymbol{\theta}})$
  $\boldsymbol{probword} : \boldsymbol{p} \leftarrow \boldsymbol{ProbDis}[:, \text{length}(\hat{\boldsymbol{x}})]$
  sample a token $y$ from distribution $\boldsymbol{q} \propto \boldsymbol{p}^{1/\tau}$
  $\hat{\boldsymbol{x}} \leftarrow [\hat{\boldsymbol{x}}, y]$
**end**
**return** $\hat{\boldsymbol{x}}$

### 2.2.5 Search Algorithm

For convenience, assume that the output of the decoder is a text string. Call the size of the output dictionary $\mathcal{Y}$ (containing all the words that can appear in the output string, including " $< eos >$ ") as $|\mathcal{Y}|$ , and the maximum length of the output string is $T'$ . Thus, a total of $\mathcal{O}(|\mathcal{Y}|^{T'})$ output sequences can be generated. All substrings after " $< eos >$ " in the output string will be

omitted. In addition, we denote $\mathbf{c}$ as the context vector that encodes the information of all hidden states from the input.

**Greedy search.** At each time step $t'$ of the output sequence, we choose the word with the highest conditional probability in $|\mathcal{Y}|$ word as output as follows:

$$y_{t'} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} P(y \mid y_1, \ldots, y_{t'-1}, \mathbf{c}) \tag{2.3}$$

When " $< eos >$ " is encountered or when the output string reaches the maximum length $T'$, we terminate the prediction. So the conditional probability of an output sequence generated from the input sequence is: $\prod_{t'=1}^{T'} P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c})$. The biggest problem with greedy search is that there is no guarantee that the found string is the optimal one.

**Exhaustive Search.** If we want the most probable sequence, we could use an exhaustive search, which involves listing all the possible output sequences and their conditional probabilities and then outputting the sequence with the highest anticipated probability. This would accomplish our goal, but at the exorbitant computing cost of $\mathcal{O}(|\mathcal{Y}|^{T'})$, where O is the base and Y and T' are the exponents, where Y and T' are the lengths of the sequence and Y are the base sizes, respectively.

**Beam search.** This is a refined version of the greedy search algorithm. The beam size, denoted by the hyper-parameter k.At the first time step, k words with the highest conditional probability are selected as the beginning of k possible sequences of output values. Based on the k candidate output sequences from the previous time step, we calculate and select the k sequences with the highest conditional probability from the total $k\,|\mathcal{Y}|$ capability at each subsequent time step. These are the output sequences that could potentially be produced at that time step. To conclude, in order to obtain the complete list of candidate output strings, we first filter out any strings in the set that contain the string " $< eos >$ " and then reject any strings that come after that character. This gives us the final set of candidate output strings.

In the final set of candidate output sequences, we will take the sequence with the highest score as the output sequence. The score for each series is calculated as follows:

$$\frac{1}{L^\alpha} \log P(y_1, \ldots, y_L) = \frac{1}{L^\alpha} \sum_{t'=1}^{L} \log P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c}), \tag{2.4}$$

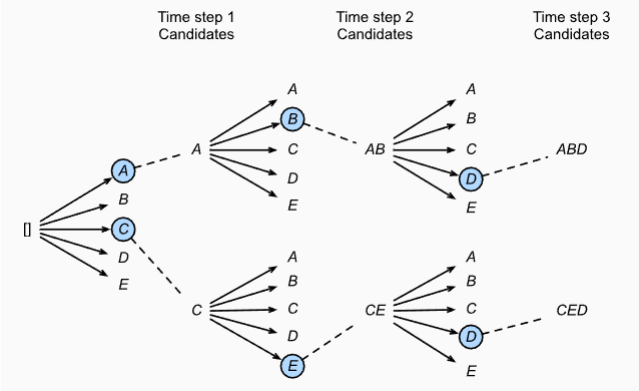


Figure 2.4: The example of beam-search.

Here, $L$ is the length of the final candidate sequence, and $\alpha$ is usually set to 0.75. $L^{\alpha}$ in the denominator is the penalty on the logarithmic sum for long strings. It can be estimated that the computational cost of the beam search is $\mathcal{O}(k\,|\mathcal{Y}|\,T')$ . It ranges between the computational cost of greedy search and exhaustive search. Alternatively, greedy search can be thought of as a beam search with a beam size of 1.By using the k beam size in a flexible manner, the beam search is able to find a compromise between the computational cost and the search quality.

## 2.2.6 Evaluate quality machine translation

### 2.2.6.1 Human Evaluate

The human-based method gives the best assessment of the quality of the translation, but this assessment is time-consuming and expensive due to the need to hire linguists to evaluate.

### 2.2.6.2 Automatic Evaluate: BLEU

BLEU stands for Bilingual Evaluation Understudy, is a method of evaluating a translation based on reference translations, the BLEU score was proposed by Kishore Papineni, *et al.* in their paper [**?**]. The prerequisite to be able to use BLEU is that you must have one (or more) sample sentences. For the machine translation problem, the sample sentence is the output sentence of a pair of sentences in the data set. BLEU evaluates a sentence by matching it with sample sentences and gives a scale from 0 (absolute deviation) to 1 (absolute match).
BLEU is known to be a simple, easy to understand, low computational cost

method similar to human evaluation. However, the human factor in sentence pattern making makes BLEU not completely objective. For example, the same sentence can have many good translations, and sometimes it is not possible to write them all in one set of sample sentences.

$$BLEU_{score} = BP \cdot exp(\sum_{i=1}^{n}(w_i log p_i)) \tag{2.5}$$

Where:
$p_i$ : average of the modified n-gram precisions using n-grams up to length N
$w_i$ : positive weights
BP (Brevity Penalty): Short penalty used to penalize translations that are too brief. The short penalty is calculated over the entire corpus:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{2.6}$$

Where:
$c$ be the length of the candidate translation.
$r$ be the effective reference corpus length.
The BLEU score can be computed by first counting the number of n-gram matches that can be found between the sample sentence (R) and the phrase being assessed (C), and then dividing that total by the total number of tokens in C. The choice of n depends on the language, task, and specific goal. The simplest we can use uni-gram is n-gram containing one token (n=1). Visually, the larger n, the smoother the sentence. This score is position-independent, so BLEU cannot evaluate word order. This is both an advantage and a limitation of BLEU. In language, a sentence can be represented by different word orders but still have to follow certain rules. In addition, to avoid having a repeated translation of a word still get a "high" score (e.g., "this this this this" versus "this is a cat"), BLEU takes into account the number of occurrences. Maximum expression of each n-gram in all sample sentences to limit the maximum number of matches. In the above example, this will only be counted once."

### 2.2.7 Reinforcement Learning

Although reinforcement learning is considered to be its own branch of machine learning, it does share certain characteristics with other types of machine learning, which can be categorized according to one of the following four domains:

- **Supervised Learning** Algorithms are trained on labeled data. The only properties that supervised learning algorithms can learn are those that are included in the data set. Image recognition models are typical supervised learning applications. These models learn to detect common characteristics of specified forms after receiving a batch of tagged photos.
- **Unsupervised learning** Developers let algorithms loose on completely unlabeled material in unsupervised learning. Without being instructed on what to look for, the algorithm learns by recording its own observations regarding data features.
- **Semisupervised learning** This strategy strikes a middle ground. Developers enter a more extensive corpus of unlabeled data along with a relatively modest sample of training data that has been labeled. The algorithm is then told to apply the knowledge it has gained from the labeled data to the unlabeled data and make inferences about the set as a whole.
- **Reinforcement learning** This adopts a completely different strategy. It places an agent in a situation with certain parameters separating useful behavior from nonbeneficial action as well as a big goal to accomplish. In that algorithms must be given clearly defined goals and defined rewards and penalties, it is akin to supervised learning in several aspects. As a result, more explicit programming is needed than in supervised learning. The algorithm, however, functions independently after these parameters are specified, making it far more self-directed than supervised learning algorithms. Due to this, reinforcement learning is occasionally referred to as a subset of semisupervised learning, although in reality, it is most frequently recognized as a distinct subset of machine learning. Refer to figure 2.5 for more illustration.

#### 2.2.7.1 Continuous and discrete actions space

In reinforcement learning, **action** is the mechanism by which the agent transitions between states of the environment. Let take an example for illustration.
**Breakout Atari game**:
The goal of the game is to eliminate as many bricks as possible with only one ball by hitting them with the walls and/or the paddle at the bottom. It's possible for the paddle to go LEFT, RIGHT, or STAY, these are **actions**.

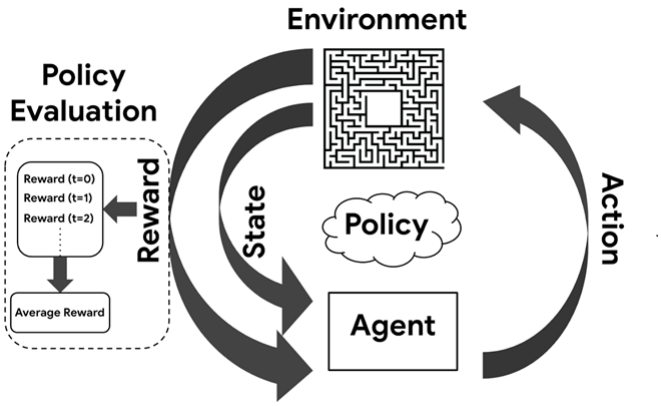


Figure 2.5: Picture showing reinforcement learning in action

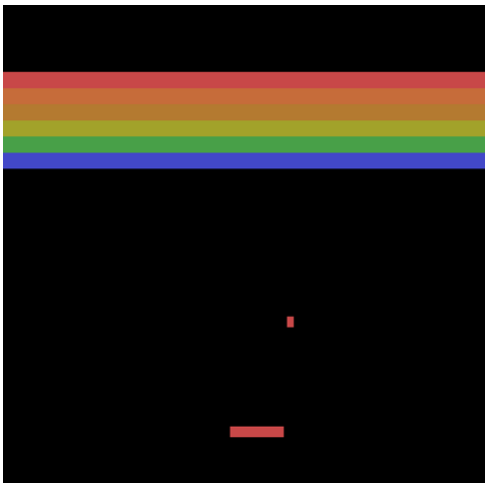

Figure 2.6: Atari game

**Discrete action space.** In the figure 2.7 we see the ball flying to the left, so we will have to move the paddle to the left to catch the ball. The discrete action mean we need to answer the question "Which direction should I move?". In this situation, we can only choose 1 of **3 ways** to move the paddle: left. In short, discrete action space is when the number of actions is **finite**.

**Continuous action space.** In the figure 2.8 we see the ball flying to the left, so we will have to move **fast** the paddle to the left to catch the ball. The discrete action mean we need to answer the question "How fast should I move?". In this situation, we need to choose to move the paddle to the left

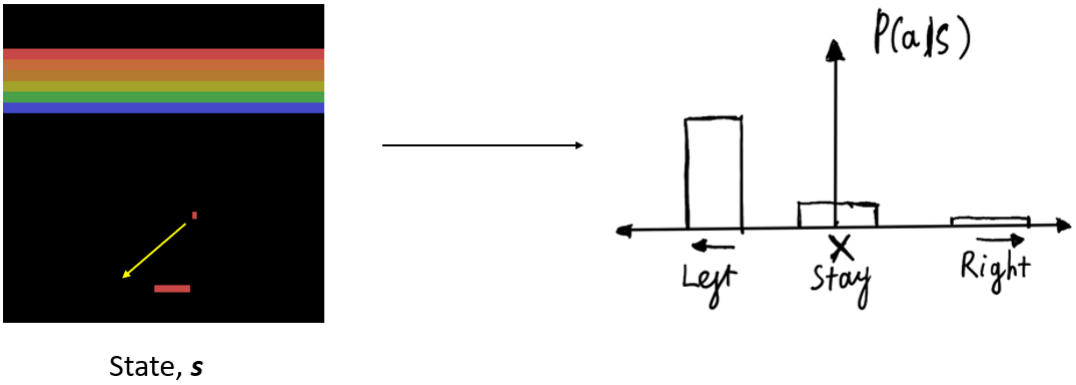


Figure 2.7: Discrete actions

with a selected speed from 0 to 10 meters per second, if it is too fast or too slow, it will not catch the ball. In short, continuous action space is when the number of actions is **infinite**. Because of the **infinite number of possible actions (Translated sentences)** so the Neural Machine Translation task is Continuous Action Space. Therefore, we only choose the Reinforcement Learning Method that can solve Continuous Action Space such as the Actor-Critic method.

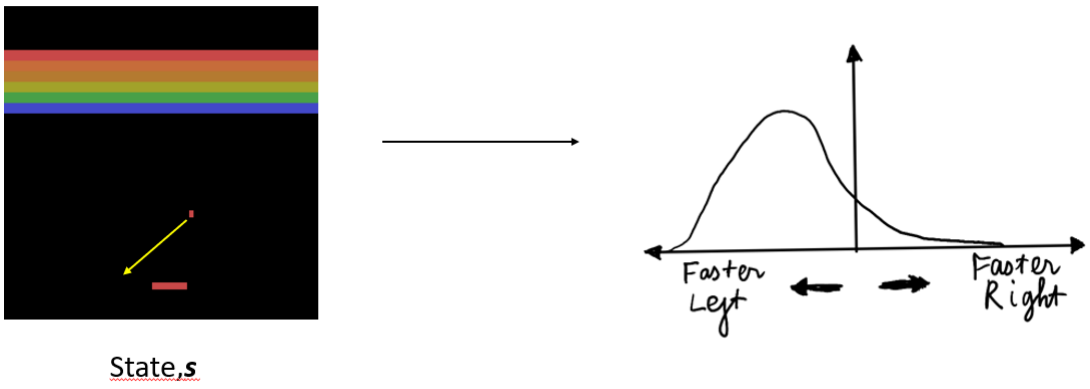


Figure 2.8: Continuous action

#### 2.2.7.2 Policy Gradient

This section talk about how to train policy gradient for continuous action space. Because there are infinitely many possible actions to choose from,

there is no computer memory that can store all of the actions so that there can be an optimal solution to the problem. So these models only give action selection probability distributions.
As an example, we have to drive a car on the road to reach our destination. Vehicles can turn left and turn right at arbitrary speeds at different times; if the vehicle is farther from the curb than the given distance, then 10 points will be added, if less than the given distance, 10 points will be deducted, and points will be 100 when the car reaches to the destination. End the game when the car hits the curb or reaches the destination. In the beginning, we drive randomly until we reach the finish line or hit the curb and save those actions in memory. Actions that increase the points increase the probability of their occurrence in the action distribution, and vice versa, actions that reduce the reward points decrease the probability of their occurrence; this is repeated until the model learns how to drive the car to the finish line without hitting the curb. Refer to figure 2.9 for more illustration.

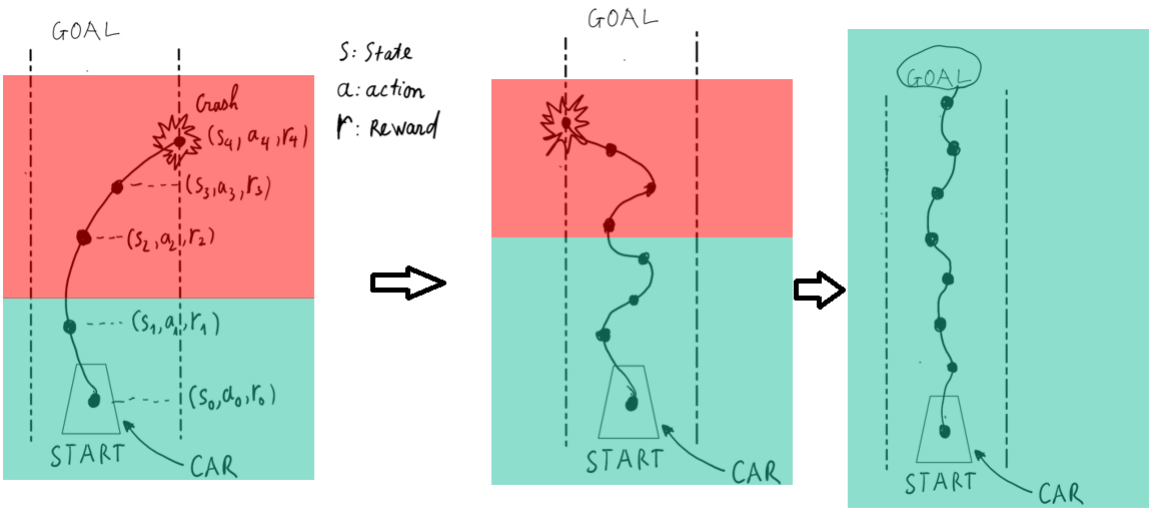


Figure 2.9: Continuous actions

In short the algorithm is:

- Initialize the agent.
- Run a policy until termination.
- Record all states, actions, rewards.
- Decrease probability of actions that resulted in low reward.
- Increase probability of actions that resulted in high reward.

#### 2.2.7.3 Actor-Critic method

- Actor-critic approaches explicitly describe the policy apart from the value function in a distinct memory structure.
- Because the policy structure is used to select actions, it is known as the "actor".
- The estimated value function is referred to as the "critic" since it critiques the actor's activities.
- The criticism is presented as a temporal difference error. This scalar signal is the only output of the critic, and it drives all "actor" and "critic" learning.
- The critic is a function of state-value. After choosing each action, the critic looks at how things turned out to see if they were better or worse than expected. That evaluation is the temporal difference error. In neural machine translation temporal difference error can be BLEU score.

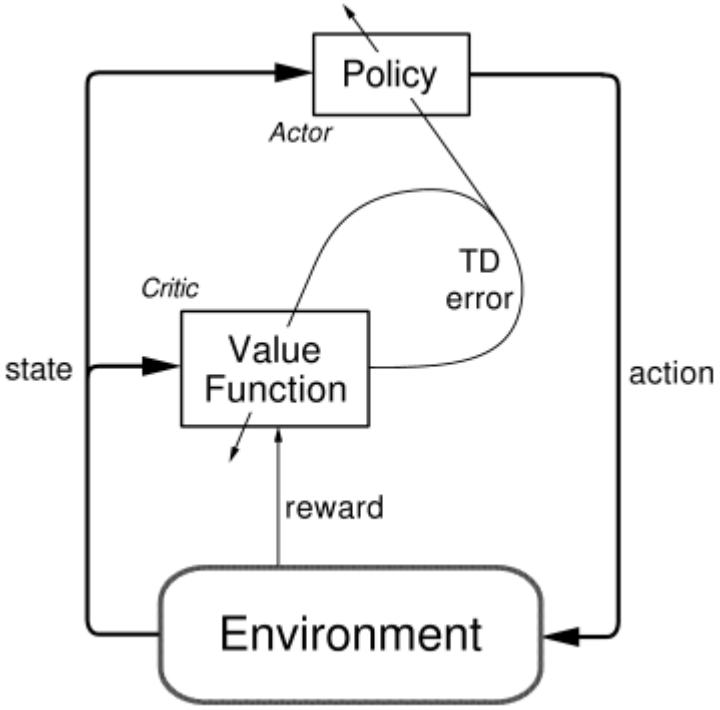


Figure 2.10: Illustration of Actor critic

# Chapter 3

# Methodology

In this chapter we will introduce how to apply actor-critic algorithm to neural machine translation.

## 3.1 RL Environment in NMT

When solving a problem for reinforcement learning, the first and most important thing is to define the environment, state, action, and reward points [10]. When there is little to learn from the new feedback, such as when it is supplied on the same data that the model was already trained on, empirical improvements over a powerful pre-trained model disappear. This is because there are no non-zero-reward translation outputs sampled when starting from a random policy. Reinforcement learning techniques offer the potential to improve machine translation models in scenarios where there are no reference translations, only reward signals, and models may be pre-trained on existing data. [4]. It means we cannot get the reward signal at training on the training set because the maximum likelihood estimation algorithm [1] has already done its job very well [11]. Therefore, we will take the signal point at inference, or that is, take the reward signal when evaluating the model on the validation set.
So we will define action, reward, environment and state as follows:

- Environment = Valid set.
- State = Source sentence
- Agent = Neural machine translation.
- Actions = Hypothesis sentence generate from agent.
- Reward = BLEU score calculate between reference sentence and Hypothesis sentence.
- State-Value function: The critic model calculates the score between the source and the hypothetical sentence. This model can be trained by instructing it according to the BLEU score. We will talk more about it in the next chapter.

To make it easier to imagine, this method is like teaching a student; the student is the agent, and the teacher is the critic. Students will learn everything in the lessons. The exercises here have answers available for students to compare, and after learning, they use the knowledge they have learned to apply to the tests; the teachers will grade them and tell students to try to do better in the next test (students do not know the answer to the question). As a result, the students will gradually get better.

## 3.2 Actor-Model: Transformer

Transformer-base is the NMT method adopted in our investigations [9]. To be more exact, there are N = 6 layers and h=8 parallel attention layers, or heads. The inner layer of feed-forward networks has a dimensionality of $d_{ffn} = 2048$, whereas the input and output have a dimensionality of $d_{model} = 512$. We follow [9] regularization and optimization techniques. We not only use byte pair encoding [8] for tokenizing training data from scratch but also use the drop-out method in the paper [12] for a better translation model. We use this toolkit Fairseq [13] to pre-train the actor model. We will train the model until they ***converge***.

The following table is the result of pre-trained for each dataset mention in 2.1.

| Data set | Vocab by BPE method | BLEU score in IWLST 2015 Test Set |
|---|---:|---:|
| IWLST 2015 | 10.000 | 28.53 Vi2En —–29.44 En2Vi |
| IWLST 2015 | 55.000 | 29.56 Vi2En —– 28.17 En2Vi |
| PhoMT | 32.000 | 44.64 Vi2En —– 39.38 En2Vi |

Table 3.1: Pre-trained Critic Models

The ***converge*** means that the BLEU score cannot improve for a long training even if we use any regularization techniques of deep learning. Example like the figure 3.1 below. We can see that after reaching a 28.17 BLEU (at about step 22k) score in test set, the model can't be better.

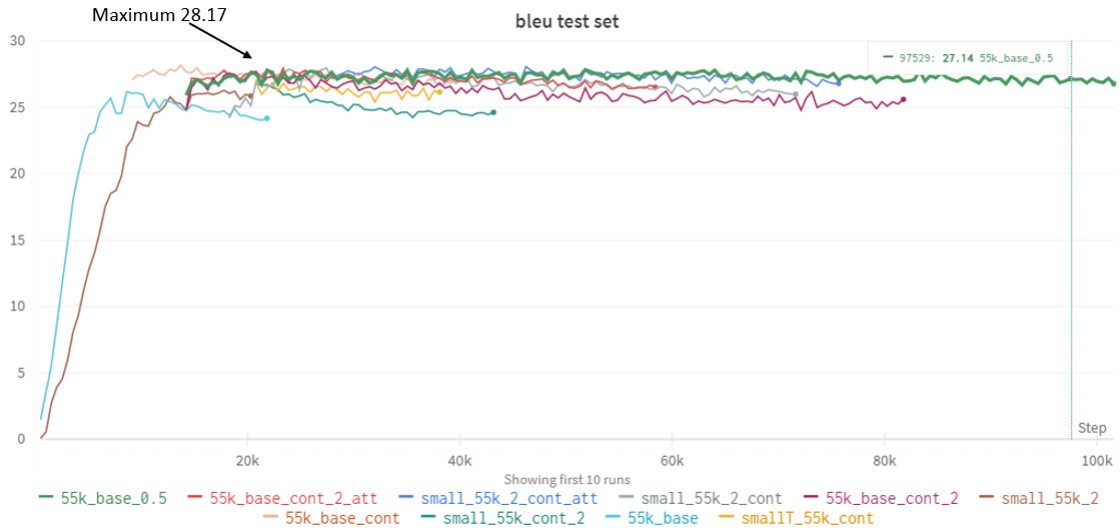


Figure 3.1: BLEU score test set during pre-training en2vi

## 3.3 Critic-Model

Critic detect whether the translated sentence is the same as the human-translated sentence. It is "State-Value function (Source X, Hypo Y)" in Actor-Critic method. Refer to figure 3.2 for illustration.

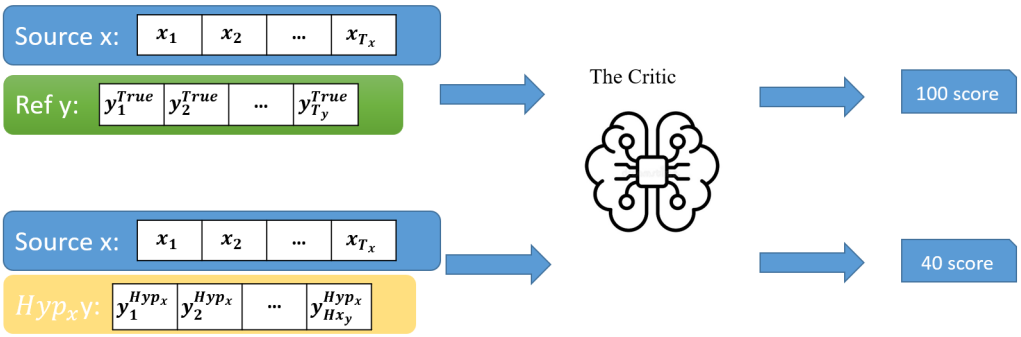


Figure 3.2: Illustration of how critic work

### 3.3.1 Convolution architecture

The critic base on convolution architecture is taken from paper by Yang *et al.* [6].Since the sentences produced by the generator have varying lengths, M

is the maximum length that can be specified for output from the generator, which is utilized to turn the sentences into sequences with fixed lengths using CNN padding. Taking into account the sequence in the source language $s_1, \ldots, s_M$ and sequence of target language $t_1, \ldots, t_M$ , we build the matrix of the source $S_{1:M}$ and the matrix of target $T_{1:M}$ respectively as:

$$S_{1:M} = s_1; s_2; \ldots; x_M \tag{3.1}$$

And

$$T_{1:M} = t_1; t_2; \ldots; t_M \tag{3.2}$$

Where $s_m, s_m \in R^k$ is the k-dimensional word embedding and the semi-colon is the concatenation operator. Then we will combine these 2 matrices into a 2d image and use convolution to calculate whether the sentence is like the human translator or not.

$$p = \sigma(convolution\,(V\,[c_s; c_t])) \tag{3.3}$$

Where $V$ is the transform matrix which transforms the concatenation of $S$ and $T$ into a 2-dimension embedding and $\sigma$ is the logistic function. Refer 3.3 for illustration.

**Training Algorithm** To train this model, we use the same algorithm in the Generative Adversarial Networks [14] by labeling the image concatenated by the source sentence and the human-translated sentence as 1, and the image concatenated by the source sentence and the translated by actor sentence is 0, this is same technique of paper by Yang *et al.* [6].

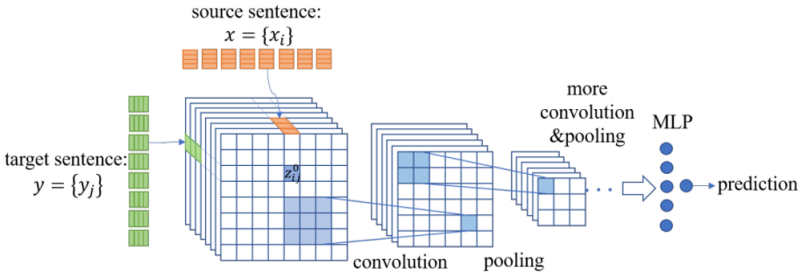


Figure 3.3: Critic base on convolution architecture

### 3.3.2 Transformer architecture

The critic base on convolution architecture is taken from paper by Lee *et al.* [5].Given a source phrase x, an NMT model will generate a set of hypotheses in the target language in the form of the mathematical expression distribution $\mathcal{H}(s) = \{h_1, h_2, \ldots, h_n\}$. This study aims to develop a critic that achieves greater scores for hypotheses of higher quality, where quality is defined in terms of a user-specified measure $\mu(u, r)$ such as BLEU. [2].

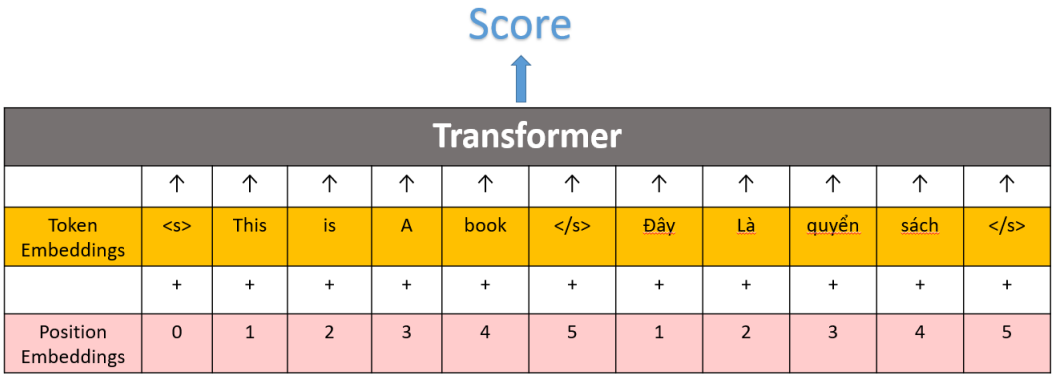


Figure 3.4: Critic base on transformer architecture

As shown in 3.4, our critic is a transformer architecture that accepts as input the combination of the source phrase s and the hypothesis $h \in \mathcal{H}(s)$. Additionally, position embeddings are a part of the design, and they give the model the ability to represent tokens that are common to both languages. The final hidden state corresponding to the begin-of-sentence token ($\langle s \rangle$) serves as the joint representation for $(s, h)$; $z \in \mathbb{R}^d$ is the notation used to refer to this feature vector. The critic assigns a scalar score $o \in R$ to $(s, h)$ by applying a one hidden layer neural network with$d_{tanh}$ hidden units to $z$, The parameters of the critic are denoted by the "classification head" of RoBERTa's basic design [15]. These parameters include the parameters of the transformer, all embeddings, and the top projection block that translates the feature vector to the scalar score. Therefore, each hypothesis $h_i$ in the collection $H(s)$ is separately processed and provides a score $o_i$ to see if this sentence is similar to the human-translator.

**Training Algorithm** By reducing the amount that the target distribution deviates from the model output distribution and minimizing the Kullback–Leibler divergence, $D_{KL}\left(p_T \| p_M\right)$, we train the critic to be discriminative [16]. The model output distribution is a softmax over the n hypotheses

that are included in the n-best list for each s.

$$p_M\left(s, h_i \mid s; \theta\right) = \frac{\exp\left(o_i\left(h_i \mid s; \theta\right)\right)}{\sum_{j=1}^{n} \exp\left(o_j\left(h_j \mid s; \theta\right)\right)} \tag{3.4}$$

Where we specified that the score $o_j$ is dependent on the input vector sx and parameter vector $\theta$. Please note that we do not impose any additional factorization. Specifically, we do not assume that the score is computed via auto-regression. The goal distribution is defined as a normalized distribution of the final metric $\mu\left(h_i, r\right)$ (user-specified metric such as BLEU score), which we expect improves as its values increase:

$$p_T\left(h_i\right) = \frac{\exp\left(\mu\left(h_i, r\right)/T\right)}{\sum_{j=1}^{n} \exp\left(\mu\left(h_j, r\right)/T\right)}, \tag{3.5}$$

Where T is the temperature used to control the distribution's smoothness. In practice, min-max normalization is used to. We start by subtracting each value from the minimum in the list of hypotheses, then we divide the resulting number by the difference between the highest value and the minimum value. This gives the best hypothesis a score of 1, while the hypothesis with the lowest score receives a score of 0. According to [17], this aids in the optimization process by reducing the gradient variance (2018). The critic parameters are then discovered by reducing the KL divergence over the training set. The following serves as a particular illustration of training:

$$\mathcal{L}(\theta) = -\sum_{j=1}^{n} p_T\left(h_j\right) \log p_M\left(h_j \mid s; \theta\right) \tag{3.6}$$

Since all terms are differentiable, we minimize this loss over the training set using stochastic gradient descent and standard backpropagation of the error. The training algorithm is same as algorithm in paper by Lee *et al.* [5].

## 3.4 Training and Inference

### 3.4.1 Training Objective

Actor A's goal is to generate, starting from the initial state, a sequence that will maximize the amount of its anticipated end reward. The formal formula for computing the objective function is as follows:

$$J(\theta) = \sum_{Y_{1:T}} A_\theta\left(Y_{1:T} \mid X\right) \cdot R_{C,Q}^{A_\theta}\left(Y_{1:T-1}, X, y_T, Y^*\right) \tag{3.7}$$

Where A is actor, $\theta$ represents the parameters in A; C is for "critic." The completion of the goal sequence is required in order to receive a reward value from C. $Y_{1:T} = y_1; y_2; \ldots; y_T$ is the generated target sentences, in this setup we use $T = 5$, X is the source-language sentence. $Y^*$ is the reference target sentence. $R^{A_\theta}_{C,Q}$ is action-value function of a target-language sentence given the source sentence X. To estimate the action-value function, we consider the estimated probability of being real by the critic and the output of the BLEU objective Q as the reward:

$$\begin{aligned} &R^{G_\theta}_{C,Q}\left(Y_{1:T-1}, X, y_T, Y^*\right) = \\ &\lambda\left(C\left(X, Y_{1:T}\right)\right) + (1-\lambda) Q\left(Y_{1:T}, Y^*\right) \end{aligned} \tag{3.8}$$

$\lambda$ is a hyper-parameter control between exploitation and exploration [10] in reinforcement learning, for simplicity we choose $\lambda = 0.7$.
For ease of understanding, this training will consist of two training models in turn. The actor first generates translated sentences from the original sentence, thereby creating a dataset to train the critic. Then, when we evaluate the valid set, we will take the BLEU rating and the critic score together to generate a signal to adjust the actor's loss function. A good return score will direct the actor to learn in that direction, while a low score will motivate the actor to learn in another direction. These 2 models will learn until they reach Nash-equilibrium. Nash Equilibrium is a scenario in game theory in which no player in a non-cooperative game has anything to gain by changing their strategy [18]. For more illustration refer to figure 3.5

### 3.4.2 Inference

After training is complete, to use the model when translating, we use beam-search as usual but instead of taking the sentence with the highest probability generated by beam search, we use the sentence with the highest score by the critic, see figure 3.6 for more details.

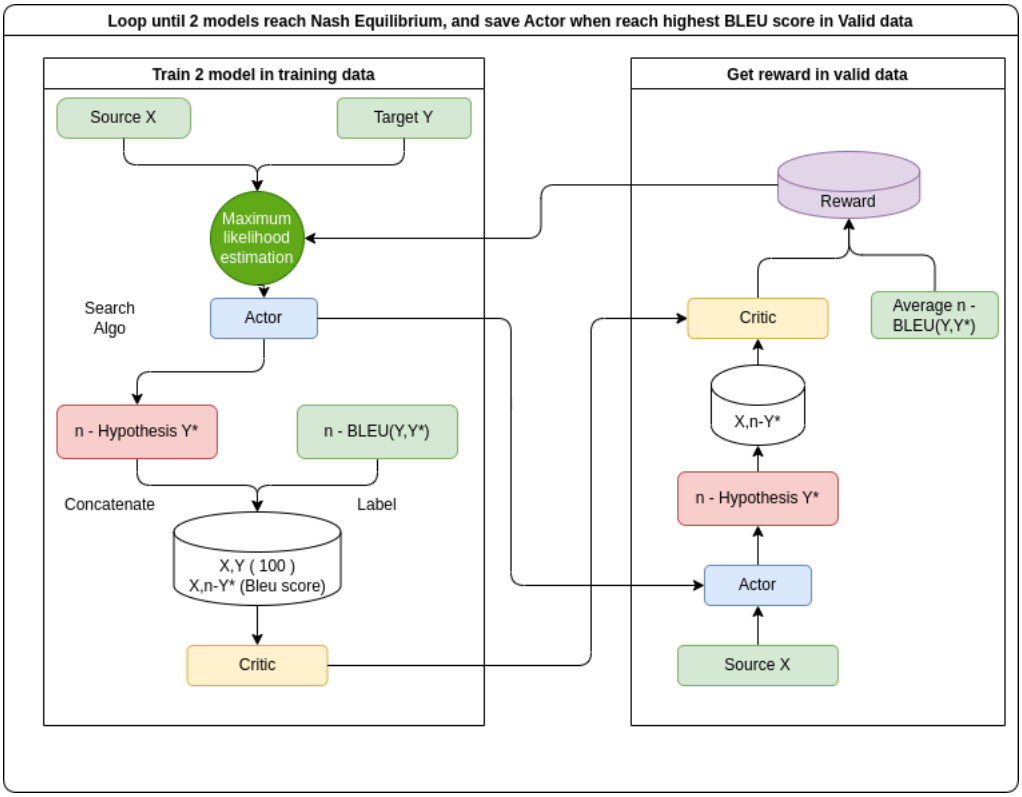


Figure 3.5: Diagrammatic representation of the system in its training mode

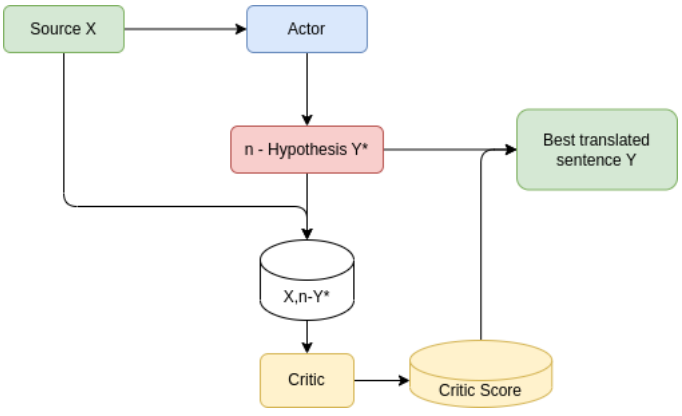


Figure 3.6: Diagrammatic representation of the system during the inference stage

# Chapter 4

# Experiment and Result

Because of limitation of hardware, we only apply full system to IWLST'15 dataset. With PhoMT dataset we just pre-train the critic for re-ranking output of actor.

## 4.1 Convolution-Critic results

From pre-train 10k vocab actor English to Vietnamese IWLST'15 with 29.44 BLEU score (refer to table 3.1 ). Refer to figure 4.1 after very long training two models reach the Nash-equilibrium we can see that model has maximum BLEU score is 28.47 and can't improve from 29.44 BLEU score in test set . So convolution architecture for critic doesn't work.

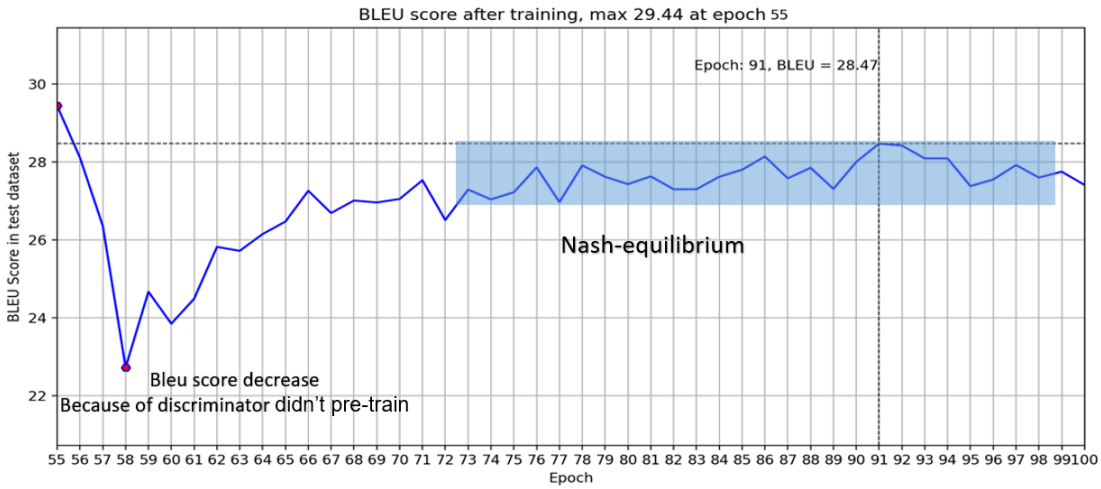


Figure 4.1: Result by CNN-critic, En2Vi BLEU score of test IWSLT'15 dataset

## 4.2 Transformer-Critic results

Based on the two tables of results 4.1, 4.2, we see that the model critic using the transformer architecture gives very good results when testing on the test set of IWSLT 2015, here we use SacreBLEU [19] with beam size = 5 for get result.

| Vocab | Base-line | Actor-critic | Δ Improve |
|---|---|---|---|
| 10k IWSLT'15 | 29.44 | 30.33 | 0.89 |
| 55k IWSLT'15 | 29.56 | 31.06 | 1.5 |

Table 4.1: Result for English to Vietnamese

## 4.3 Frozen actor, re-rank by critic.

Due to the fact that the PhoMT dataset is so large, it is approximately 22 times larger than that of IWLST'15, while the training time of 1 epoch for the small dataset is approximately 3 hours on the A100 GPU. Therefore, to demonstrate the effective method, we will only pre-train the actor with the PhoMT dataset and then use the training data of IWSLT'15 to create a dataset for the critic training. After the training is over, use this critic to evaluate the actor's translated sentences on the test set and take the sentence with the highest critic score to calculate the BLEU score. This experiment only use transformer architecture for critic model. Refer to figure 4.2.

At full setting, in each step of each input sentence, the actor will create 5 hypothesis sentences 3.4.1, so to be objective with each input sentence in the training set IWLST'15, we will sample 50 hypothetical sentences. Therefore, the dataset to train the critic look like 4.3.

| Vocab | Base-line | Actor-critic | Improve |
|---|---|---|---|
| 10k IWSLT'15 | 28.53 | 29.52 | 0.99 |
| 55k IWSLT'15 | 28.17 | 31.55 | 3.38 |

Table 4.2: Result for Vietnamese to English

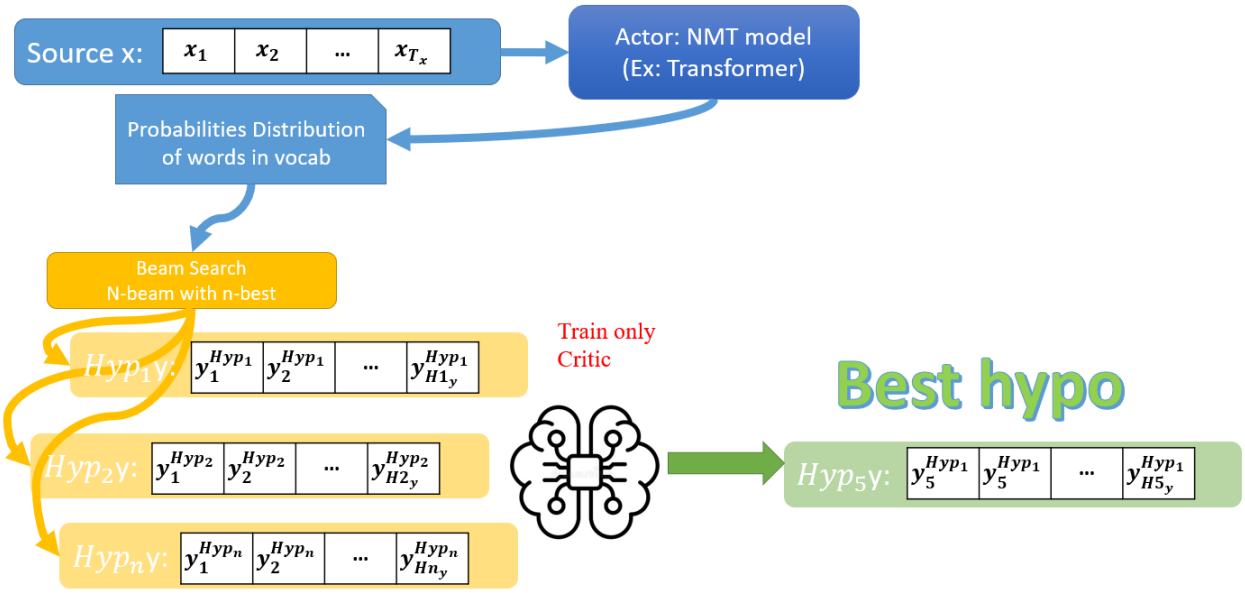


Figure 4.2: Frozen the Actor and get best translated by only critic

| IWSLT'15 English-Vietnamese data | Train | Valid | Test |
|---|---|---|---|
| Beam size: 50<br>N-best: 50 | 133317*50=6,665,850 | 1,553 | 1,268 |

Figure 4.3: Dataset for training only critic

As we can see in table 4.3 the critic transfomer-architecture perform good in translation from Vietnamese to English IWLST'15 dataset, and have competitive result with current state-of-the-art model with extra training data SOTA in IWLST'15.

| | Ennglish to Vietnamese | Vietnamese to English |
|---|---|---|
| 32k PhoMT base | 39.38 | 44.65 |
| Re-rank by critic | 38.06 $\Delta = -1.32$ | **44.80** $\Delta = 0.15$ |
| SOTA with extra training data | **40.2** | 43.6 |

Table 4.3: Result re-ranking by only critic in IWLST'15 dataset

# Chapter 5

# Conclusion

In this study, we have applied actor-critic method in reinforcement learning for machine translation to improve machine translation results, and our propose method can fill the gap between training and inference. As we have seen this method is like a regularization in deep learning, but instead of doing it manually it is an automated method. As we can see the results in the experiment, when the vocabulary is high, it is equivalent to the number of possible actions of the actor more even if the pre-train is worse, but after applying this method, it also gives better result, see table 4.2.

For future work, as mentioned in the paper by Wu *et al.* [11] that "Reinforcement learning is still beneficial when the data is monolingual." As seen in the experiment, the critic can filter out the well-translated sentences from the actor, so if we feed the monolingual data and create the bilingual data, and then use them to further train the model, there is a good chance that we can get even better results.

# References


[1] F. W. Scholz., *Maximum Likelihood Estimation, Encyclopedia of statistical sciences.* Wiley, 1985.

[2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: https://aclanthology.org/P02-1040

[3] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, "Minimum risk training for neural machine translation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1683–1692. [Online]. Available: https://aclanthology.org/P16-1159

[4] S. Kiegeland and J. Kreutzer, "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*. Online: Association for Computational Linguistics, Jun. 2021, pp. 1673–1681. [Online]. Available: https://aclanthology.org/2021.naacl-main.133

[5] A. Lee, M. Auli, and M. Ranzato, "Discriminative reranking for neural machine translation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 7250–7264. [Online]. Available: https://aclanthology.org/2021.acl-long.563

[6] Z. Yang, W. Chen, F. Wang, and B. Xu, "Improving neural machine translation with conditional sequence generative adversarial nets," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana:

Association for Computational Linguistics, Jun. 2018, pp. 1346–1355. [Online]. Available: https://aclanthology.org/N18-1122

[7] L. Doan, L. T. Nguyen, N. L. Tran, T. Hoang, and D. Q. Nguyen, "PhoMT: A high-quality and large-scale benchmark dataset for Vietnamese-English machine translation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4495–4503. [Online]. Available: https://aclanthology.org/2021.emnlp-main.369

[8] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://aclanthology.org/P16-1162

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[10] A. G. B. Richard S. Sutton, *Reinforcement Learning: An Introduction 2nd*. Cambridge, UK: MIT Press, 2018.

[11] L. Wu, F. Tian, T. Qin, J. Lai, and T.-Y. Liu, "A study of reinforcement learning for neural machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3612–3621. [Online]. Available: https://aclanthology.org/D18-1397

[12] I. Provilkov, D. Emelianenko, and E. Voita, "BPE-dropout: Simple and effective subword regularization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 1882–1892. [Online]. Available: https://aclanthology.org/2020.acl-main.170

[13] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 48–53. [Online]. Available: https://aclanthology.org/N19-4009

[14] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[15] Y. Liu, L. Zhou, Y. Wang, Y. Zhao, J. Zhang, and C. Zong, "A comparable study on model averaging, ensembling and reranking in nmt," in *Natural Language Processing and Chinese Computing*, M. Zhang, V. Ng, D. Zhao, S. Li, and H. Zan, Eds. Cham: Springer International Publishing, 2018, pp. 299–308.

[16] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 129–136. [Online]. Available: https://doi.org/10.1145/1273496.1273513

[17] S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato, "Classical structured prediction losses for sequence to sequence learning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 355–364. [Online]. Available: https://aclanthology.org/N18-1033

[18] J. Nash, "Non-cooperative games," *The Annals of Mathematics, Second Series*, vol. 54, no. 2, pp. 286–295, Sep., 1951.

[19] M. Post, "A call for clarity in reporting BLEU scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 186–191. [Online]. Available: https://aclanthology.org/W18-6319