INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 16, Issue: 4, Month: August, Year: 2021
Article Number: 4217, https://doi.org/10.15837/ijccc.2021.4.4217

CCC Publications

AGORA UNIVERSITY PRESS

Regression Loss in Transformer-based Supervised Neural Machine Translation

D.X. Li, Z.Y. Luo

Dongxing Li
School of Artificial Intelligence,
Beijing Normal University, Beijing 100875, China
lidx@bnu.edu.cn

Zuying Luo*
School of Artificial Intelligence,
Beijing Normal University, Beijing 100875, China
*Corresponding author: luozy@bnu.edu.cn

Abstract

Transformer-based model has achieved human-level performance in supervised neural machine translation (SNMT), much better than the models based on recurrent neural networks (RNNs) or convolutional neural network (CNN). The original Transformer-based model is trained through maximum likelihood estimation (MLE), which regards the machine translation task as a multi-label classification problem and takes the sum of the cross entropy loss of all the target tokens as the loss function. However, this model assumes that token generation is partially independent, without realizing that tokens are the components of a sequence. To solve the problem, this paper proposes a semantic regression loss for Transformer training, treating the generated sequence as a global. Upon finding that the semantic difference is proportional to candidate-reference distance, the authors considered the machine translation problem as a multi-task problem, and took the linear combination of cross entropy loss and semantic regression loss as the overall loss function. The semantic regression loss was proved to significantly enhance SNMT performance, with a slight reduction in convergence speed.

Keywords: supervised neural machine translation (SNMT), Transformer, attention mechanism, semantic regression loss, evaluation metric.

1 Introduction

Relying on sequence-to-sequence deep neural networks (DNNs), supervised neural machine translation (SNMT) aims to automatically convert a sequence from one language to another, with true sequence pairs as inputs [12, 31]. The most prevalent SNMT approach employs the encoder-to-decoder structure, which encodes the source sequence into a context representation in one neural network and generates the target sequence in the other [4]. Notably, the two neural networks are trained simultaneously in an end-to-end fashion. In addition, the current neural machine translation (NMT) systems mostly adopt the attention mechanism [1, 12, 23, 32].

There are generally three model architectures to train the NMT neural network. The earliest architecture is recurrent neural network (RNN), which faces problems like vanishing gradients, exploding gradients, and long-range dependency. To solve the first two problems, many improved RNNs have been designed, including long short-term memory (LSTM) proposed by Hochreiter and Schmidhuber [14], gated recurrent units (GRU) proposed by Cho et al. [4], and bidirectional LSTM (BiLSTM) proposed by Schuster and Paliwal [27]. To address the long-range dependency, convolutional neural network (CNN) has been introduced by Gehring et al. [11, 12], in which a succession of convolutional layers captures the dependency of a few tokens (phrases), and concatenates the local dependency representations as the sequence representation. Recent years has seen the emergence of a novel and competitive model called Transformer for NMT [32]. Solely based on attention mechanism, Transformer uses a self-attention network (SAN) to compute the mutual relationship scores of all the tokens within the source sequence or the target sequence. Hassan et al. proved that Transformer can achieve human-level performance on some languages [13]. Therefore, Transformer-based architectures have been widely used in the field of NMT [20, 32, 33].

The NMT performance is mainly affected by the following factors: network architecture, optimization algorithm, loss function, and evaluation metric. The original Transformer uses the Adam optimizer, cross entropy loss function, and the metric of bilingual evaluation understudy (BLEU) [26]. This Transformer-based model is trained through maximum likelihood estimation (MLE) to learn the conditional probability distribution of the target token step by step, which can be regarded as a token-level target [29]. However, the model training only focuses on the loss of the target token, yet hardly pays attention to the semantic loss of the global generated sequence. Besides, the sum of the probability distribution loss is calculated under the assumption that token generation is partially independent, without realizing that tokens are the components of a sequence. Therefore, it is reasonable to include the semantic regression loss of the global target sequence in training. To disclose sentence-level influences, most scholars resorted to ensemble learning approaches to improve translation quality [5, 24, 34], such as bag-of-words (BOW) model [24], and machine learning (ML) [5]. The simple BOW model only pays attention to token frequency, failing to consider token order and sequence semantics. Meanwhile, ML models like supervised ML used by Cohn and Goodman [5], and reinforcement learning (RL) used by Wu et al. [34] would greatly complicate Transformer training.

This paper improves the Transformer as the basic learning network. Compared with the original Transformer proposed by Vaswani et al. [32], the improved Transformer has the following unique features: (1) the optimizer is AdamW proposed by Loshchilov and Hutter [22]. (2) the evaluation metrics are translation edit rate (TER) proposed by Snover et al. [30], metric for evaluation of translation with explicit ordering (METEOR) proposed by Banerjee and Lavie [2], recall-oriented understudy for gisting evaluation-the longest common subsequence (ROUGE-L) proposed by Lin [21] as well as BLEU [26]. (3) the loss function innovatively computes the semantic loss of the global sequence between the candidate and the reference, except for the cross entropy loss of all the point-wise tokens, that is, the NMT problem is treated as a regression problem from the perspective of the global sequence, in addition to the multi-label classification problem from the perspective of the tokens.

The improved Transformer was compared with the original Transformer through NMT experiments on three evaluation datasets. On dataset IWSLT2014 DE->EN, the improved Transformer outperformed the original Transformer by 0.55 BLEU/2.07 TER/0.20 ROUGE-L, and by 0.49 BLEU/2.70 TER/0.60 ROUGE-L, with transformer-small configuration and transformer-base configuration, respectively. On dataset IWSLT2016 DE->EN, the improved Transformer outperformed the original Transformer by 0.51 BLEU/-0.66 TER/0.78 METEOR/0.68 ROUGE-L, and by 0.56 BLEU/-0.07 TER/0.37 METEOR/0.87 ROUGE-L, with transformer-small configuration and transformer-base configuration, respectively. On dataset WMT17 EN->DE, the improved Transformer outperformed the original Transformer by 0.86 BLEU/-1.08 TER/0.68 METEOR/0.80 ROUGE-L, and by 1.21 BLEU/-1.14 TER/0.62 METEOR/0.83 ROUGE-L, with transformer-base configuration and transformer-big configuration, respectively.

The remainder of this paper is organized as follows: Chapter 2 introduces the preliminaries of this work, including three SNMT architectures, attention mechanisms, and training strategy, with particular emphasis on Transformer attention mechanisms; Chapter 3 puts forward our model, highlighting the realization of the novel loss function; Chapter 4 carries out a series of experiments, explains the selection of some hyper-parameters, and describes the implementation of our model; Chapter 5 summarizes our work and looks forward to future research.

2 Preliminaries

2.1 Transformer architecture

Transformer is implemented entirely based on the attention mechanism, whose maximum path length and minimum number of sequential operations are and . Consisting of stacked encoder layers and decoder layers, this model architecture can capture long-range dependencies more directly than RNNs and CNN. An encoder layer contains a multi-head self-attention layer, followed by a position-wise feedforward layer. Both layers are added with a residual connection layer and a layer normalization layer. Multi-head self-attention is implemented by multiple SANs via a linear transformation [35]. The layers of an encoder can be summarized in sequence as: self-attention -> residual connection -> layer normalization -> feed-forward -> residual connection -> layer normalization. The encoder layer can the hidden representations of all the tokens in the source sequence. A decoder layer has a similar structure to the encoder, except that an encoder-decoder attention layer is inserted, which is followed by the multi-head self-attention layer. The insertion aims to compute the attention score between the hidden representations of the source sequence and the target token representation. The layers of a decoder can be summarized in sequence as: self-attention -> residual connection -> layer normalization -> encoder-decoder attention -> residual connection -> layer normalization -> feed-forward -> residual connection -> layer normalization.

Transformer combines the merits and eliminates the defects of RNNs-based and CNN-based NMTs. It relies on an encoder multi-head self-attention network and a decoder multi-head self-attention network to capture the token dependencies in the source sequence and the target sequence, respectively. However, the SAN sequence fails to consider the token order. To remedy this issue, the relative or absolute position of the tokens in the sequence are added to the corresponding token embeddings via trigonometric functions.

2.2 Attention mechanisms in Transformer

Self-attention. Transformer architecture entirely relies on two attention mechanisms: self-attention and encoder-decoder attention. During training, Transformer employs the SAN to compute the attention scores between one token and another within the encoder and the decoder, respectively. Notably, the decoder computes the attention scores between the current token and its previous tokens, masking the subsequent tokens. Meanwhile, it applies encoder-decoder attention to compute the attention scores between the target token and all the source tokens. These attention mechanisms operate on an input sequence, $X = (x_1, x_2, \cdots x_n)$, where n is its length and $x_i \in \mathbb{R}^{d_x}$ is the $i-th$ token.The goal is to obtain a new feature matrix $O = (o_1, o_2, \cdots o_n)$ of the same dimension as X, where $o_i \in \mathbb{R}^{d_o}$ is the token representation of x_i. Here, d_x and d_o represent the dimensions of the input and latent representation, respectively, which are usually equal to the model dimension. Then, the $i-th$ hidden embedding o_i can be calculated by:

$$\begin{aligned} o_i &= \sum_{j=1}^{n} \alpha_{ij}(x_j W^V) \\ \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \\ e_{ij} &= \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_o}} \end{aligned} \tag{1}$$

where, $W^Q \in \mathbb{R}^{d_x \times d_q}, W^K \in \mathbb{R}^{d_x \times d_k}, W^V \in \mathbb{R}^{d_x \times d_v}$ $(d_q = d_k)$ are learned parameter matrices; e_{ij} is the scaled dot-product attention score; α_{ij} is attention score obtained by a softmax function subject to $\sum_{j=1}^{n} \alpha_{ij} = 1$. Similarly, the representation matrix at the sequence level can be obtained by:

$$\begin{aligned} O &= soft\max(\frac{QK^T}{\sqrt{d_o}} + MASK)V \\ Q &= XW^Q, K = XW^K, V = XW^V \end{aligned} \tag{2}$$

where, O is the attention head computed on three matrices $Q \in \mathbb{R}^{n\times d_q}, K \in \mathbb{R}^{n\times d_q}, V \in \mathbb{R}^{n\times d_v}$ whose dimensions are d_q, d_q and d_v ($d_q = d_v = d_o = d_{model}$), respectively. These matrices are packed together by queries, keys, and values which constitute the input. For encoder, $MASK \in \mathbb{R}^{n\times n}$ is named as padding mask, and used to align a batch of examples. The elements of the matrix are $0s$ or $-\infty s$, which correspond with non-padding or padding elements in X, respectively. $0s$ indicate that the tokens can attend to each other, and $-\infty s$ mean the otherwise. For decoder, $MASK \in \mathbb{R}^{m\times m}$ is a triangular matrix whose elements are all $0s$ below the diagonal and all $-\infty s$ in other places. This means that the target token can attend to its previous tokens and cannot attend to the subsequent tokens.

Encoder-Decoder Attention. This attention mainly computes the representation of the target token in decoder. On the top layer of the encoders, there are three feature matrices of the source sequence $X - Q \in \mathbb{R}^{n\times d_q}, K \in \mathbb{R}^{n\times d_q}, V \in \mathbb{R}^{n\times d_v}$. Let m be the length of the target sequence X'; $Q' \in \mathbb{R}^{m\times d_q}, K' \in \mathbb{R}^{m\times d_q}, V' \in \mathbb{R}^{m\times d_v}$ be the hidden states. Then, the final hidden state for the generated sequence can be calculated by:

$$O' = soft\max(\frac{Q'K^T}{\sqrt{d_o}} + MASK)V \tag{3}$$

where, the row vector of $MASK \in \mathbb{R}^{m\times n}$ is same as that of the padding mask in the encoder.

Multi-head Self-attention. Transformer actually employs attention heads in the original implementation. This multi-head mechanism has a great advantage: the model is allowed to capture the token representations in different vector subspaces. Many experiments have shown that different heads can also capture different language information [32]. Some subspaces include syntax information and some include semantic information. Under the multi-head mechanism, Q, K, V are first split into h, each head is represented, and all the representations are concatenated as the final output:

$$\begin{aligned} &Q_i, K_i, V_i = split(Q, K, V) \\ &head_i = Attention(Q_i, K_i, V_i) \\ &MultiHead(Q, K, V) = Concat(head_1, head_2, \cdots, head_h)W^o \end{aligned} \tag{4}$$

2.3 Model training strategy

Token-level learning. In the original Transformer architecture, NMT aims to maximize the mapping score of a sequence pair or maximize the conditional probability $\hat{y} = \arg\max_y p(y|x)$, where x and y are the source and target sequences, respectively. This task is typically regarded as a multi-label classification problem, where the class size equals the size of the target language vocabulary V. The MLE is usually adopted to address this problem:

$$\zeta_1 = -\sum_{j=1}^{n} \sum_{y_j \in V} p_{data} \log p_{model} \tag{5}$$

where, n is the length of the target sequence; V is the target vocabulary; p_{data} and p_{model} are the probability distribution of the ground-truth data with label smoothing, and the probability distribution of the model output, respectively.

Sequence-level learning. Ignoring the global sequence-level semantic loss, the token-level learning approach only computes the isolated token-level loss, which merely focuses on the cross entropy of each token. The novel sequence-level method adds BOW as another target loss, and assumes that the sequence-level probability of each token is independent of the position in the sequence [24]. Despite its simplicity and improved performance, BOW only takes account of token frequency, without considering token order and sequence meaning. Hence, the reference is not easily exposed due to the highly similarity of the candidate. Some ML models like supervised ML and RL have also been integrated into the Transformer model [5, 34]. For example, Cohn and Goodman built Bayesian model on Transformer to reduce meaning loss, applied rational speech acts (RSA) model to produce speakers and listeners which can be modeled as Bayesian agents, and utilized the role of speakers and listeners as double-sided mirrors to understand the overall sequence information [5]. The RL rewards and punishes the target translation sequence by setting an effective reward function [34]. The problem is supervised ML and RL algorithms inevitably complicates Transformer training. To solve the problem, our method adopts a simple structure and considers the semantic loss between the candidate and the reference.

3 Modeling

This chapter explains the details on our model. Besides presenting the input/output notations and training network architecture, the authors expounded at length on the implementation of semantic regression loss during training.

3.1 Notations

Some notations are necessary to facilitate the model description. For an SNMT task, the $i-th$ sample (x_i, y_i) consists of a source language sequence x_i and a target language sequence y_i. In the model architecture, x_i relates to the encoder input $x_{enc\_in(i)}$, and y_i relates to the decoder input $y_{dec\_in(i)}$ and the decoder output $y_{dec\_out(i)}$ which have 1-position misalignment. The semantic output of y_i is a vector denoted as $y_{sem(i)}$. All inputs and outputs of our model can be formatted as:

$$\begin{aligned} x_{enc\_in(i)} &= \{x_i^1, x_i^2, ..., x_i^m\} \\ y_{dec\_in(i)} &= \{y_i^1, y_i^2, ..., y_i^n\} \\ y_{dec\_out(i)} &= \{y_i^1, y_i^2, ..., y_i^n\} \\ y_{sem\_out(i)} &= y_i^{sm} \end{aligned} \tag{6}$$

where, m and n are the length of the source sequence and the target sequence respectively; sm is the dimension of the semantic vector of the generated sequence, which equals the target vocabulary size $|V|$. Despite being the same in length, the decoder input and decoder output differ due to the malposition prediction caused by the language model; $x_{enc\_in(i)}$ and $y_{dec\_out(i)}$ are the ground truth data; $y_{sem\_out(i)}$ is the sum of all the label smoothing vectors.

3.2 Network architecture

As shown in Figure 1, our model firstly preprocesses all the samples. The first step is to insert $\langle /s \rangle$ (end of the sequence) into the last position of the source sequence, and then to insert $\langle s \rangle$ (start of the sequence) and $\langle /s \rangle$ into the first position and last position of the target sequence, respectively. Before a preprocessed sample is imported into the encoder, all the token embeddings are initialized to compute their absolute positional encodings. Next, token embeddings and positional encodings are added to the encoder, and regarded as the real input of the encoder.

$$\begin{aligned} x_{enc\_in(i,j)} &= we(x_i^j) + pe(x_i^j) \\ y_{dec\_in(i,j)} &= we(y_i^j) + pe(y_i^j) \\ x_{enc\_in(i)} &= concat(x_{enc\_in(i,j)}) \\ y_{dec\_in(i)} &= concat(y_{dec\_in(i,j)}) \\ x_{enc\_out(i)} &= encoders(x_{enc\_in(i)}) \end{aligned} \tag{7}$$

where, i is the $i-th$ sample and j is the $j-th$ token. Through nonlinear computing via the stacked encoder layers, it is possible to obtain three latent representation matrices of the source sequence-Q, K, V. Then, matrices K and V are transferred into the encoder-decoder attention layer of all

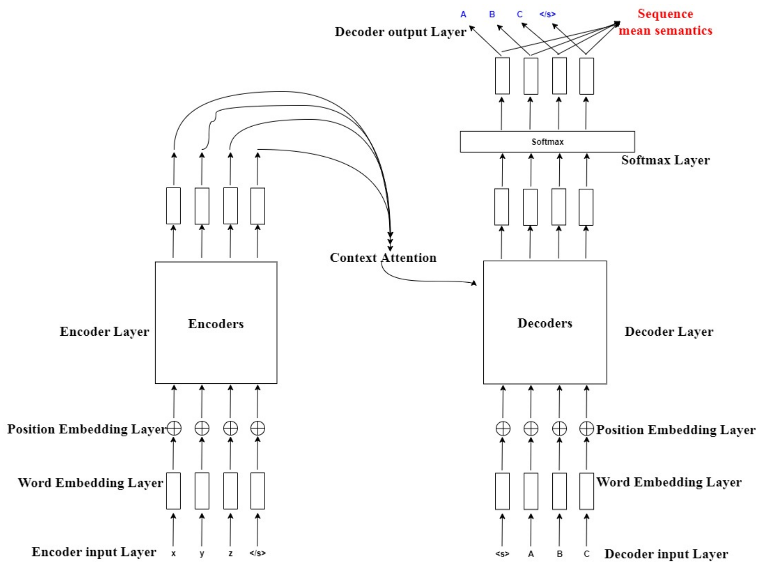

Figure 1: Architecture of our model. Note: The red part is the semantics of the candidate translation, and the other part is the original Transformer

decoder layers. Similarly, the stacked decoders are utilized to generate the probability distribution of the target token, conditioned on the encoder output and previous generated tokens, as well as the masked attention scores.

$$\begin{aligned} &token_i^t = decoders(xenc\_out(i), y_{dec\_in(i)}^{[:t]}) \\ &score_i^t = W \times token_i^t + b \\ &P(token_i^t) = soft\max(score_i^t) \end{aligned} \tag{8}$$

Through nonlinear transform of the stacked decoders, the token probability distribution at $t-th$ step is obtained by softmax function. The dimension of probability distribution p_j equals the target vocabulary size $|V|$ and subjects to the constraint condition $\sum_{j=1}^{n} p_{ij} = 1$. So far, learning the target token distribution has been treated as a multiple-label classification problem. Additionally, the authors analyzed the semantic distribution of the target sequence. Because these token distributions contain positional encodings, the weighted sum of them is taken as the semantics of the translation sequence.

$$\begin{aligned} &s_i^t = \textstyle\sum_{j=1}^{|V|} p_j \otimes we_j \\ &y_{sem\_out(i)} = \sum_{t=1}^{n} W^t s_i^t \end{aligned} \tag{9}$$

where, s_i^t is the semantics of the $t-th$ reference token; we_j is the token embedding; n is the length of the target sequence. Therefore, $y_{sem\_out(i)}$ can be regarded as the semantic of the generated sequence. For simplicity, the average of the weighted sum $\frac{1}{n}\sum_{j=1}^{n} s_i^t = 1$ is applied as the approximate value of $y_{sem\_out(i)}$.

3.3 Semantic regression loss function

At the $t-th$ step, the original Transformer considers the distribution of the target output is regarded as a multi-label classification problem. Let n and $|V|$ be the length of the target sequence and the size of the target vocabulary, respectively. The mean sum of token cross entropy is taken as the first loss, which is computed by formula (5) and denoted as ζ_1. To improve the translation quality during the iterative training, the meaning of the candidate translation should be close to the reference sequence, and the sematic distance between the best choice and the reference should be minimized as in a Procrustes problem. For a given vocabulary on a specific dataset, it is possible to obtain the token embeddings, and denote them as a matrix $E_v \in \mathbb{R}^{|V|\times d}$, where $|V|$ is vocabulary size and d

is the dimension of token embedding. This matrix is always relatively static. In other words, the token embedding changes slightly with corpuses and vocabularies. For a sequence, the mean sum of all the token embeddings is taken as its eigenvector. Therefore, a target sequence can be expressed as a one-hot encoding $s_{sen} \in \mathbb{R}^{|V|}$ without taking label smoothing into account, e.g., $[1, 0, 0, 0, 1, \cdots\ 0, 1]$. Here, the authors compute the multiplication between one component of $p_{i,j}$ and $\vec{e_j}$, and then sum up the matrix on $axis = 0$, treating the transpose of the vector as the approximate semantics of the target sequence:

$$\begin{aligned} M_{sen} &= \begin{bmatrix} p_{11} & p_{21} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ p_{1|V|} & p_{2|V|} & \cdots & p_{n|V|} \end{bmatrix} \\ E_v &= \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1d} \\ e_{21} & e_{22} & \cdots & e_{2d} \\ \cdots & \cdots & \cdots & \cdots \\ e_{|V|1} & e_{|V|2} & \cdots & e_{|V|d} \end{bmatrix} = \begin{bmatrix} \vec{e}_1 \\ \vec{e}_2 \\ \cdots \\ \vec{e}_{|V|} \end{bmatrix} \end{aligned} \tag{10}$$

where, the column vector in $M_{sen} \in \mathbb{R}^{|V|\times n}$ is the target token probability distribution; $E_v \in \mathbb{R}^{|V|\times d}$ is the target embedding lookup table, in which each row vector is the embedding corresponding to each token in the target vocabulary. Then, the latent representation of the target token, e.g., the column vector of s_{token}, can be obtained in the matrix form by formula (11), and the semantics of the sequence can calculated by adding up the multiplications between M_{sen} by formula (11).

$$\begin{aligned} s_{token} &= \begin{bmatrix} p_{11}\vec{e}_1 & p_{21}\vec{e}_1 & \cdots & p_{n1}\vec{e}_1 \\ p_{12}\vec{e}_2 & p_{22}\vec{e}_2 & \cdots & p_{n2}\vec{e}_2 \\ \cdots & \cdots & \cdots & \cdots \\ p_{1|V|}\vec{e}_{|V|} & p_{2|V|}\vec{e}_{|V|} & \cdots & p_{n|V|}\vec{e}_{|V|} \end{bmatrix} \\ s_{sen} &= \sum_{axis=0} M_{sen}^T \otimes E_v \\ &= \sum_{axis=0} (\begin{bmatrix} p_{11} & p_{21} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ p_{1|V|} & p_{2|V|} & \cdots & p_{n|V|} \end{bmatrix}^T \otimes \begin{bmatrix} \vec{e}_1 \\ \vec{e}_2 \\ \cdots \\ \vec{e}_{|V|} \end{bmatrix}) \\ &= \sum_{axis=0} (\begin{bmatrix} p_{11}\vec{e}_1 + p_{21}\vec{e}_1 + \ldots + p_{n1}\vec{e}_1 \\ p_{12}\vec{e}_2 + p_{22}\vec{e}_2 + \ldots + p_{n2}\vec{e}_2 \\ \cdots \\ p_{1|V|}\vec{e}_{|V|} + p_{2|V|}\vec{e}_{|V|} + \ldots + p_{n|V|}\vec{e}_{|V|} \end{bmatrix}) \end{aligned} \tag{11}$$

where, s_{sen} is the convex combination of the semantic embeddings of the target sequence [25]. In this way, it is possible to acquire the semantics of the predicted sequence and target sequence. The semantic regression loss can be derived from the distance between the semantics of the two sequences. Furthermore, the target tokens are mapped into the same d-dimensional vector space, where the vectors are not highly variable. Hence, the semantic distance is approximately taken as the second loss from the global sequence:

$$\begin{aligned} &D_{s(y,\hat{y})} = s_{sen} - \hat{s}_{sen} = M_{sen} \otimes E_v - \hat{M}_{sen} \otimes E_v = (M_{sen} - \hat{M}_{sen}) \otimes E_v \propto (M_{sen} - \hat{M}_{sen}) \\ &D_{s(y,\hat{y})} \approx (M_{sen} - \hat{M}_{sen}) \\ &\zeta_2 = \left| D_{s(y,\hat{y})} \right| \end{aligned} \tag{12}$$

where, y and $\hat{y}$ are the reference sequence and candidate sequence, respectively; S_{sen} and $\hat{S}_{sen}$ are the semantics of the reference sequence and candidate sequence, respectively. If the vector space of tokens does not change, the semantic difference between the reference and the candidate must be proportional to the vector distance.

In the original Transformer, the decoder is trained in the teacher forcing way, that is, the translation system actually knows the correct answer, when it predicts the subsequent token. However, the decoder works in an autoregressive manner during the inference. In this way, the token does not know what the subsequent token is. To bridge the gap, it is necessary to make an overall analysis of the semantic differences between the reference sequence and the candidate sequence. Therefore, the decoder could be trained from the perspectives of the token and the global target sequence simultaneously. The first loss aims to maximize the probability distribution of the generated token, while the second loss aims to maximize the similarity between the reference and the candidate, and to preserve the meaning. To weight the difference, scalar λ is introduced to balance the loss functions as a hyper-parameter.

$$\begin{aligned} &\zeta = \lambda\zeta_1 + (1-\lambda)\zeta_2 \\ &s.t. \ \ \lambda \in [0,1] \end{aligned} \tag{13}$$

4 Experiments

This chapter verifies the performance of our model through experiments. Firstly, the sequence pairs of three evaluation datasets (Section 4.1) were preprocessed. Then, the experimental setting, evaluation metrics (Section 4.2) and baseline systems based on Transformer (Section 4.3) were introduced in turn. Finally, experimental implementation was detailed, including hyper-parameter selection (Section 4.4) and results analysis (Section 4.5).

4.1 Datasets

Our experiments were conducted on three translation tasks: IWSLT 2014/2016 DE->EN translation dataset and WMT17 EN->DE translation dataset, which are widely used as evaluation benchmarks for NMT [32]. Each dataset was split into a training set, a validation set, and a test set. Then, the data were preprocessed through normalization and sub-word segmentation. The IWSLT dataset contains the data extracted from the IWSLT Evaluation Campaign [3, 9]. IWSLT2014 offers 160k/7k sentence pairs as training/validation sets. The authors concatenated dev2010, dev2012, tst2010, tst2011 and tst2012 as the test set, including about 7k sentence pairs. For IWSLT2016, the data consist of 180k/12k sentence pairs as training/validation sets. The authors took the concatenation of tst2010/2011/2012/2013/2014 as the test set, including about 12k sentence pairs. For WMT17, the original training set was adopted as the training set of our model, which contains 5.9 million sentence pairs. The concatenation of newstest2013/2014/2015/2016 was taken as the validation set, involving more than 1 million sentence pairs, and newstest2017 was treated as the test set, containing about 3k sentence pairs (see Table 1).

The three datasets above were preprocessed by Moses, a de-facto standard toolkit for SMT [16]. Firstly, the sentence pairs were tokenized to deal with the punctuations. Considering the huge range of tokens in the tokenized sequences, the sequence pairs of more than 100/80 tokens were discarded for IWSLT2014/2016 and WMT17, respectively, such as to improve translation quality and accelerate training. Finally, every subset of each dataset was truncated. To mitigate the influence of out of vocabulary (OOV) tokens and rare tokens, the sequence was tokenized to sub-word units by SentencePiece ([39]) [18] or WordPiece ([40]) [17], which have the same function of bytes pair encoding (BPE) [28]. In addition, each subset shares a vocabulary of 32,000, because English and German belong to the Germanic language family.

4.2 Experimental setting and evaluation metrics

The implementation of our experiments is based on the empirical results. Our experiments are build using appropriate setting, such as, optimizer, learning rate and other hyper-parameters. For optimizer, we choose AdamW optimizer [22] with $weight\_decay = 10^{-5}, \beta_1 = 0.9, \beta_2 = 0.999$ which is used in Bert [6]. For learning rate, we also follow the learning rate warm-up strategy [32] with

Table 1: The sequence pair statistics after preprocessing by Moses ((a)) and subword statistics after segmentation by BPE ((b)) on IWSLT 2014/2016 DE->EN translation dataset and WMT17 EN->DE translation dataset. The terms of "Train", "Eval" and "Test" represent training set, validation set and test set respectively. "S" and "T" represent the source language and the target language respectively. Note: units of k and m stand for thousand and million respectively

Dataset	Train	Eval	Test
IWSLT2014	161k	7.3k	6.7k
IWSLT2016	181k	12k	11.8k
WMT17	5.85m	1.12m	3k

(a)

Dataset	S/T	Train	Eval	Test
IWSLT 2014	DE	4.13m	188k	167k
	EN	4.06m	184k	-
IWSLT 2016	DE	4.65m	306k	290k
	EN	4.59m	302k	-
WMT17	EN	155.36m	423k	94k
	DE	169.71m	460k	-

(b)

$warmup - steps = 8000$. During training, label smoothing rate is set to 0.1 and all dropout values are set to 0.1. Our experiments were implemented under the framework of TensorFlow 1.14.0. The attainted model cpt files also can be easily converted into PyTorch bin files using transformers. All the experiments were completed by two NVIDIA 1080Ti GPUs. During the inference, beam search size was set to 4 for validation set and test set.

There are many evaluation metrics, each of which has its own strengths and weaknesses. For diversity and reliability, four metrics were selected to evaluate our model, namely, BLEU [26], METEOR [2], TER [30] and ROUGE-L [21]. The authors computed BLEU score with standard Moses tools of multi-bleu.perl ([36]), and TER score with pyter ([38]), and METEOR score, ROUGE-L score with nlg-eval ([37]). BLEU, as the earliest automatic evaluation method for machine translation, analyzes the degree of co-occurrence of n-gram between the candidate and the reference. The main component of BLEU is n-gram precision via geometric averaging. METEOR is based on explicit word-to-word matches, which includes the identical words in the surface forms, morphological variants in stemmed forms, and synonyms in meanings between the candidate and the reference. This metric combines unigram-precision, unigram-recall, and a direct measure of how out-of-order the words in the candidate translation are. TER is a distance-based metric of the workload of the post-translation editing of the candidate translation. The distance is defined as the minimum number of edits which transforms one sequence into another. TER considers edit operations like insertion, deletion, and substitution of single words, as well as shifts of word sequences. ROUGE is a recall-based metric commonly used for machine translation and text summarization. Chin-Yew Lin introduced four different ROUGE measures: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. Here, ROUGE-L is chosen as the metric to evaluate machine translation. L stands for the longest common subsequence (LCS) of the corresponding sequences of the candidate and the reference.

4.3 Baseline systems of machine translation based on the original Transformer

Our model was compared to the original Transformer models with small, base, and big configurations. The difference between our model and the original Transformer lies in the application of the semantic regression loss during training. Drawing on Vaswani et al.'s work [32], the transformer-base model and transformer-large model were designed to contain 6 layers and 12 layers, respectively. Facing the limited GPU resources, the number of stacked encoders and decoders was set to 6 in all the configurations for our model. Each transformer (small, base, and big) has a unique configuration. The small, base, and big configurations differ in the number of heads, hidden size, filter size, and number of blocks (layers) of encoders and decoders. The corresponding hyper-parameters of the three models are listed in Table 2. According to the different sizes of the three datasets, transformer-small and transformer-base settings were adopted for IWSLT DE->EN, and transformer-base and transformer-big settings for WMT17 EN->DE.

Table 2: The hyper-parameters of transformer-small, transformer-base and transformer-big configurations of Transformer based machine translation systems

Transformer	Heads	Blocks	Hidden size	Filter size
small	4	6	256	1024
base	8	6	512	2048
big	16	6	1024	4096

4.4 SNMT based on our model

1) Different distance measurement algorithms

The candidate sequence and the reference sequence were separately taken as a whole. In the computing graph of the network architecture, the two sequences are essentially two tensors or multidimensional vectors, and their distance is denoted as ζ_2 . In this section, our model was tested with three different metrics for the distance between the two vectors—x, y, where x is the semantic representation of the candidate sequence and y is the semantic representation of the reference sequence: Euclidean distance (square): $ED(x, y) = \|x - y\|_2^2 = (x - y)^T(x - y)$; Cosine distance (cosine similarity): $Cos(x, y) = x \cdot y = x^T y = y^T x$. Note that the cosine distance is generalized without normalization, which is equivalent to the dot product operation, without affecting the final result; Max-pooling distance (MPD): $MPD(x, y) = \max(x_i, y_i)|_1^n$. Inspired by max-pooling, this metric maximizes the corresponding components of the two vectors. Our model was implemented on the three datasets with transformer-base configuration. Some experimental results are listed in Table 3.

Multiple trials and comparisons were carried out to contrast the cross entropy loss of each token with Kullback–Leibler divergence (KLD, [19]) and Jensen-Shannon divergence (JSD, [8]), both of which are distribution metrics. Under these circumstances, the metric scores were slightly improved. However, cross entropy (CE) is the simplest algorithm in terms of computing. From the left side of Table 3, it can be seen that the use of CE brings stable effects. Therefore, the experiments on the right side of Table 3 all use the CE algorithm. From the right side of Table 3, it was inferred that ED > MPD > Cos achieved apparently different performances, with the conventional ED being the best performer. Moreover, MPD had a similar effect as ED. These experiments were implemented with formula (13). Overall, the linear combination of CE and ED was determined as our loss function.

Table 3: The different evaluation scores using different distance evaluation algorithms on IWSLT2014, IWSLT2016 and WMT17 with transformer-base configuration respectively. CE, KLD [19] and JSD [8] are ζ_1 loss which stand for cross-entropy, Kullback–Leibler divergence and Jensen-Shannon divergence respectively. ED, Cos and MPD are ζ_2 loss which stand for Euclidean distance, Cosine distance and Max-pooling distance respectively. ED+CE, Cos+CE and MPD+CE are the linear combinations between CE and different distance losses

Loss		CE	KLD	JSD	ED+CE	CE+CE	KLD+CE
IWSLT2014	BLEU	33.96	33.90	**33.97**	**34.45**	34.32	34.21
	TER	**50.71**	50.97	50.98	**48.01**	48.15	49.07
	METEOR	**33.73**	32.77	33.29	33.07	33.71	**33.80**
	ROUGE_L	62.95	63.04	**63.26**	**63.55**	63.47	63.11
IWSLT2016	BLEU	**32.45**	32.40	32.41	33.01	**34.32**	34.21
	TER	51.38	**51.37**	50.48	51.31	**48.15**	49.07
	METEOR	**32.52**	31.07	31.29	32.89	33.71	**33.80**
	ROUGE_L	**60.23**	59.04	59.26	61.10	**63.47**	63.11
WMT17	BLEU	28.10	28.25	**28.29**	**28.96**	28.41	28.66
	TER	55.11	**54.97**	54.98	**54.03**	54.76	54.49
	METEOR	28.73	**28.75**	28.71	**29.41**	28.58	28.87
	ROUGE_L	56.22	56.24	**56.26**	**57.02**	56.43	56.68

2) Effect of hyper-parameter λ

In order to weight the two loss functions, our model was tested with the λ, value changing from 0.1 to 0.9 with a step length of 0.2. As shown in Table 4, the four evaluation scores obeyed the normal distribution, peaking at $\lambda = 0.5$. Hence, distance loss is as important as cross entropy loss for our model. In terms of performance, the semantics loss of the global sequence is as important as the cross entropy loss of all the tokens. Moreover, in the decoder, each step is a multi-label classification problem, from the perspective of token translation. From the perspective of the global sequence translation, however, each step is a regression task. The results in Table 4 show that $\lambda = 0.5$ is the best choice for our model.

Table 4: The different evaluation scores using different λ on IWSLT2014, IWSLT2016 and WMT17 with transformer-base configuration respectively. $\lambda = 1$ means that only ζ_1 is used. All results are obtained by using ED+CE

λ		0.1	0.3	0.5	0.7	0.9	1
IWSLT2014	BLEU	33.98	34.13	**34.45**	34.28	33.80	33.96
	TER	49.76	48.38	**48.01**	49.30	49.92	50.71
	METEOR	32.89	32.51	33.07	33.51	33.53	**33.73**
	ROUGE_L	63.47	63.43	63.55	**63.80**	62.84	62.95
IWSLT2016	BLEU	32.73	32.54	**33.01**	32.15	32.37	32.45
	TER	53.09	52.28	51.31	**51.30**	52.95	51.38
	METEOR	30.98	31.63	**32.89**	31.42	32.53	32.52
	ROUGE_L	59.37	60.33	**61.10**	59.19	60.09	60.23
WMT17	BLEU	27.73	28.54	**28.96**	28.41	28.53	28.10
	TER	55.99	55.28	**54.03**	57.30	55.95	55.11
	METEOR	27.89	28.51	**29.41**	28.51	28.53	28.73
	ROUGE_L	55.17	55.93	**57.02**	55.80	55.84	56.22

3) Effects of adam optimizer cluster

The next is to test the impact of different Adam optimization algorithms on machine translation performance. The authors focused on comparing three optimization algorithms, namely, Adam [15], NAdam [7] and AdamW [22], especially Adam and AdamW. The Adam optimizer integrates traditional momentum with RMSProp. NAdam optimizer combines Nesterov accelerated gradient (NAG) with Adam, replacing the traditional momentum in the original Adam with Nesterov momentum [7]. AdamW optimizer was adopted for our model, which targets Adam's roller coaster problem. To avoid overfitting, Adam optimizer employs L2 to update weights. But this approach is susceptible to large weight decay. Hence, the weight decay method in the Adam algorithm should be adopted instead of L2 regularization. The experimental results in Table 5 demonstrate the effectiveness of AdamW optimizer. Compared to the Adam optimizer, AdamW achieved a performance improvement of 0.38 BLEU/0.56 TER/1.42 METEOR/0.79 ROUGE-L on the IWSLT2014 dataset, 0.34 BLEU/1.05 TER/0.40 METEOR/1.09 ROUGE-L on the IWSLT2016 dataset, and 0.26 BLEU/0.45 TER/0.30 METEOR/0.85 ROUGE-L on the WMT17 dataset.

4) Effect of beam size

To test its performance more accurately, our model was compared with a beam search algorithm [10], using different beam sizes during inference. Beam=1 means to use greedy search for machine translation. As shown in Table 6, the results of greedy search were far worse than those of beam search algorithm. In addition, $Beam = 3$ brought significant improvement and could be seen as an inflection point during inference. Further, little fluctuation occurred when the beam size was greater than 3. To save time and space, Beam=4 was usually the best choice. Compared to greedy search, the performance improvement was 3.19 BLEU/5.31 TER/2.60 METEOR/3.26 ROUGE-L on the IWSLT2014 dataset at Beam=4, 1.78 BLEU/3.90 TER/3.62 METEOR/3.63 ROUGE-L on the IWSLT2016 dataset at Beam=4, and 1.79 BLEU/4.09 TER/2.15 METEOR/3.14 ROUGE-L on the WMT17 dataset at Beam=4.

5) Experimental results

The experiments (1)-(4) help to identify the key hyper-parameters that ensure the success of our

Table 5: The different evaluation scores using different optimizers on IWSLT2014, IWSLT2016 and WMT17 with transformer-base configuration respectively. All results are obtained by using ED+CE and $\lambda = 0.5$

Optimizer		Adam	Nadam	AdamW
IWSLT2014	BLEU	34.07	34.28	**34.45 (+0.38)**
	TER	48.57	48.32	**48.01 (-0.56)**
	METEOR	31.65	32.24	**33.07 (+1.42)**
	ROUGE_L	62.76	62.81	**63.55 (+0.79)**
IWSLT2016	BLEU	32.67	32.62	**33.01 (+0.34)**
	TER	52.36	52.53	**51.31 (-1.05)**
	METEOR	32.49	32.48	**32.89 (+0.40)**
	ROUGE_L	60.01	59.81	**61.10 (+1.09)**
WMT17	BLEU	28.70	28.73	**28.96 (+0.26)**
	TER	54.48	54.90	**54.03 (-0.45)**
	METEOR	29.11	28.89	**29.41 (+0.30)**
	ROUGE_L	56.17	56.17	**57.02 (+0.85)**

Table 6: The different evaluation scores using different beam sizes on IWSLT2014, IWSLT2016 and WMT17 with transformer-base configuration respectively. All results are obtained by using ED+CE, $\lambda = 0.5$ and AdamW optimizer

Beam size		1	2	3	4	5	6	7	8	9	10
IWSLT2014	BLEU	31.26	32.74	34.12	**34.45**	33.41	32.89	32.94	33.74	33.94	34.09
	TER	53.31	52.03	49.51	48.01	**47.95**	48.03	49.32	49.25	48.23	48.54
	METEOR	30.47	31.60	31.95	**33.07**	32.64	31.85	31.93	32.32	32.71	32.92
	ROUGE_L	60.29	61.66	62.17	63.55	62.35	61.75	62.27	62.48	63.31	**63.68**
IWSLT2016	BLEU	31.23	31.75	32.40	33.01	32.54	32.18	31.87	32.45	**33.10**	32.66
	TER	55.21	54.13	53.81	**51.31**	52.39	51.93	53.14	53.78	51.56	51.82
	METEOR	29.27	31.28	31.46	**32.89**	31.27	31.28	30.43	31.82	32.57	32.32
	ROUGE_L	57.47	58.28	58.98	**61.10**	58.88	56.49	57.45	57.49	58.99	59.72
WMT17	BLEU	27.17	27.74	28.35	**28.96**	28.49	27.17	27.74	28.35	28.70	28.49
	TER	58.12	56.11	55.01	**54.03**	54.93	58.03	56.02	55.02	54.87	54.95
	METEOR	27.26	27.30	28.75	**29.41**	28.87	27.27	27.39	28.79	29.11	29.01
	ROUGE_L	53.88	54.23	55.38	**57.02**	56.00	53.92	54.23	55.49	56.17	55.97

model. The overall results of our experiments are shown in Table 7. All metric scores were computed on the condition of uncased tokens. For IWSLT2014/2016 DE->EN translation tasks, our model was tested on test sets. As shown in Table 7, our approach improved the performance over the baseline system using the transformer-small and transformer-base configurations, respectively. The performances on IWSLT2014/IWSLT2016 were improved by 0.55 BLEU/-2.07 TER/0.20 ROUGE-L and 0.51 BLEU/-0.66 TER/0.78 METEOR/0.68 ROUGE-L with transformer-small configuration, respectively. The performances on IWSLT2014/IWSLT2016 were improved by 0.49 BLEU/-2.70 TER/0.60 ROUGE-L and 0.56 BLEU/-0.07 TER/0.37 METEOR/0.87 ROUGE-L with transformer-base configuration, respectively. The slight improvement may relate to the size of the dataset. For WMT17 EN->DE translation tasks, our model achieved progressive improvement when using transformer-big configuration. For the transformer-base configuration, our model outperformed the baseline system by 0.86/-1.08/0.68/0.80 using BLEU, TER, METEOR and ROUGE-L, respectively. For transformer-big configuration, our model outperformed the baseline system by 1.21/-1.14/0.62/0.83 using BLEU, TER, METEOR and ROUGE-L, respectively. These performances were achieved under the setting of ED+CE, $\lambda = 0.5$, AdamW optimizer and Beam=4.

6) Training time cost

Further, the time cost of the original Transformer was compared with our model. The following

Table 7: The overall evaluation scores whether using the semantics distance loss on IWSLT2014, IWSLT2016 and WMT17 test set by using ED+CE, $\lambda = 0.5$, AdamW optimizer and Beam=4

Models	Metrics	IWSLT2014	IWSLT2016	WMT17
Transformer-small	BLUE	33.78	32.17	-
	TER	51.77	52.83	-
	METEOR	33.29	31.93	-
	ROUGE_L	62.91	60.05	-
Transformer-base	BLUE	33.96	32.45	28.10
	TER	50.71	51.38	55.11
	METEOR	33.73	32.52	28.73
	ROUGE_L	62.95	60.23	56.22
Transformer-big	BLUE	-	-	29.08
	TER	-	-	54.37
	METEOR	-	-	29.09
	ROUGE_L	-	-	56.78
Transformer-small+ours	BLEU	**34.33 (+0.55)**	**32.68 (+0.51)**	-
	TER	**49.70 (-2.07)**	**52.17 (-0.66)**	-
	METEOR	32.60	**32.71 (+0.78)**	-
	ROUGE_L	**63.11 (+0.20)**	**60.73 (+0.68)**	-
Transformer-base+ours	BLEU	**34.45 (+0.49)**	**33.01 (+0.56)**	**28.96 (+0.86)**
	TER	**48.01 (-2.70)**	**51.31 (-0.07)**	**54.03 (-1.08)**
	METEOR	33.07	**32.89 (+0.37)**	**29.41 (+0.68)**
	ROUGE_L	**63.55 (+0.60)**	**61.10 (+0.87)**	**57.02 (+0.80)**
Transformer-big+ours	BLEU	-	-	**30.29 (+1.21)**
	TER	-	-	**53.23 (-1.14)**
	METEOR	-	-	**29.71 (+0.62)**
	ROUGE_L	-	-	**57.61 (+0.83)**

can be inferred from the results in Table 8. Horizontal comparison: the convergence speed is negatively correlated with dataset size. Meanwhile, the time to reach convergence is positively correlated with dataset size. Longitudinal comparison: our model consumed more time than the original Transformer, especially for the datasets with small size. Compared with WMT17, it cost more time for IWSLT2014/2016 training with our model.

4.5 Discussion

This paper proves that the semantic distance loss is as important as cross entropy loss between the translation output and the reference during training. The translation quality can be improved with the linear combination of the two losses as the overall loss, at the cost of a slight increment in time overhead. As in the original Transformer, the generation task in the decoder essentially solves a multi-label classification problem from the perspective of token generation. This paper also focuses on the semantic distance between the generated sequence and the reference sequence from the perspective of sequence generation.

There are three possible reasons for the insignificant improvement on performance: (1) The limited improvement over IWSLT14 and IWSLT16 comes from the certain degree of overfitting due to the small training set. (2) The semantics of a sequence are more than the average meaning of the tokens. Each token has a unique weight on the semantics of the sequence. Thus, the parameters in formula (9) should be learned during model training. (3) At the start of training, there is not much prior knowledge about the generation of the target sequence. Thus, it is more suitable to use ζ_1 as the loss function for this stage. With the growing number of iterations in training, it is better to adopt the linear combination of cross entropy loss and semantic regression loss as the loss function. However, this paper implements the linear combination from the very beginning, failing to determine when is the best time to start using this loss function. Hence, the future research will try to determine when

Table 8: The time costs on IWSLT2014, IWSLT2016 and WMT17 train set. Note: units of s, h and d stand for second, hour and day respectively

Models	Time	IWSLT2014	IWSLT2016	WMT17
Transformer-small	Total	7h	9h	-
	Step	0.064s	0.065s	-
Transformer-base	Total	10h	15h	11.8d
	Step	0.188s	0.189s	0.205s
Transformer-big	Total	-	-	23.8d
	Step	-	-	0.435s
Transformer-small+ours	Total	**9h**	**11h**	-
	Step	**0.068s**	**0.067s**	-
Transformer-base+ours	Total	**12.5h**	**16.5h**	**12.5d**
	Step	**0.197s**	**0.194s**	**0.212s**
Transformer-big+ours	Total	-	-	**23.9d**
	Step	-	-	**0.438s**

to replace the loss function.

5 Conclusions

This paper proposes a novel semantic regression loss function for the SNMT based on the Transformer architecture. The authors find that the semantic loss function is proportional to the distance between the reference sequence and the candidate sequence, conditioned on the given target language dataset and vocabulary. Hence, the linear combination of the cross entropy loss (classification objective function) and the distance loss (regression objective function) are synthetized as our training objective function. Then, our model is implemented on three datasets, and evaluated by four metrics. During the experiments, our model is coupled with the Euclidean distance metric, AdamW optimizer and beam search algorithm. The results show that our model can effectively improve the machine translation performance.

However, the semantic loss proposed here is not actually a true semantic loss. It is approximately equivalent to the distance loss between the candidate sequence and the reference sequence. In addition, the linear combination of ζ_1 and ζ_2 is implemented from the beginning to the end, which limits the performance improvement. In future work, the authors will use a pre-training language model like Bert to compute the exact semantic loss in the decoder, apply prior knowledge to the loss function, and determine the golden section points for choosing the different losses.

Funding

This work was supported by the National Natural Science Foundation of China under Grant 61977009.

Author contributions

The authors contributed equally to this work.

Conflict of interest

The authors declare no conflict of interest.

References

[1] Bahdanau, D.; Cho, K.; Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473*, 2014.

[2] Banerjee, S.; Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments, In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* 65-72, 2005.

[3] Cettolo, M.; Niehues, J.; Stüker, S.; Bentivogli, L.; Federico, M. (2014). Report on the 11th iwslt evaluation campaign, In *Proceedings of the International Workshop on Spoken Language Translation*, Hanoi, 2014.

[4] Cho, K.; Van Merriënboer, B.; Gulcehre, C.;Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv:1406.1078*, 2014.

[5] Cohn-Gordon, R.; Goodman, N. (2019). Lost in machine translation: A method to reduce meaning loss, *arXiv preprint arXiv:1902.09514*, 2019.

[6] Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805*, 2018.

[7] Dozat, T. (2016). Incorporating nesterov momentum into adam, *ICLR 2016 workshop homepage 2016*, 2016.

[8] Endres, D.M.; Schindelin, J.E. (2003). A new metric for probability distributions, *IEEE Transactions on Information theory*, 49(7): 1858-1860, 2003.

[9] Federmann, C.; Lewis, W.D. (2016). Microsoft speech language translation (mslt) corpus: The iwslt 2016 release for english, french and german, In *International Workshop on Spoken Language Translation*, 2016.

[10] Freitag, M.; Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation, *arXiv preprint arXiv:1702.01806*, 2017.

[11] Gehring, J.; Auli, M.; Grangier, D.; Dauphin, Y.N. (2016). A convolutional encoder model for neural machine translation, *arXiv preprint arXiv:1611.02344*, 2016.

[12] Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. (2017). Convolutional sequence to sequence learning, In *International Conference on Machine Learning*, 1243-1252, 2017.

[13] Hassan, H.; Aue, A.; Chen, C.; Chowdhary, V.; Clark, J.; Federmann, C. (2018). Achieving human parity on automatic chinese to english news translation, *arXiv preprint arXiv:1803.05567*, 2018.

[14] Hochreiter, S.; Schmidhuber, J. (1997). Long short-term memory, *Neural computation*, 9(8): 1735-1780, 1997.

[15] Kingma, D.P.; Ba, J. (2014). Adam: a method for stochastic optimization, *CoRR abs/1412.6980*, 2014.

[16] Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N. (2007). Moses: Open source toolkit for statistical machine translation, In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, 177-180, 2007.

[17] Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates, *arXiv preprint arXiv:1804.10959*, 2018.

[18] Kudo, T.; Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, *arXiv preprint arXiv:1808.06226*, 2018.

[19] Kullback. S.; Leibler, R.A. (1951). On information and sufficiency, *The annals of mathematical statistics*, 22(1): 79-86, 1951.

[20] Li, Y.; Wang, Q.; Xiao, T.; Liu, T.; Zhu, J. (2020). Neural machine translation with joint representation, In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(5): 8285-8292, 2020.

[21] Lin, C.Y. (2004). Rouge: A package for automatic evaluation of summaries, *In Text summarization branches out*, 74-81, 2004

[22] Loshchilov, I.; Hutter, F. (2017). Decoupled weight decay regularization, *arXiv preprint arXiv:1711.05101*, 2017.

[23] Luong, M.T.; Pham, H.; Manning, C.D. (2015). Effective approaches to attention-based neural machine translation, *arXiv preprint arXiv:1508.04025*, 2015.

[24] Ma, S.; Sun, X.; Wang, Y.; Lin, J. (2018). Bag-of-words as target for neural machine translation, *arXiv preprint arXiv:1805.04871*, 2018.

[25] Norouzi, M.; Mikolov, T.; Bengio, S.; Singer, Y.; Shlens, J.; Frome, A. (2013). Zero-shot learning by convex combination of semantic embeddings, *arXiv preprint arXiv:1312.5650*, 2013.

[26] Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. (2002). BLEU: a method for automatic evaluation of machine translation, In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311-318, 2002.

[27] Schuster, M.; Paliwal, K.K. (1997). Bidirectional recurrent neural networks, *IEEE transactions on Signal Processing*, 45(11): 2673-2681

[28] Sennrich, R.; Haddow, B.; Birch, A. (2015). Neural machine translation of rare words with subword units, *arXiv preprint arXiv:1508.07909*, 2015.

[29] Shen, S.; Cheng, Y.; He, Z.; He, W.; Wu, H.; Sun, M.; Liu, Y. (2015). Minimum risk training for neural machine translation, *arXiv preprint arXiv:1512.02433*, 2015.

[30] Snover, M.; Dorr, B.; Schwartz, R.; Micciulla, L.; Makhoul, J. (2006). A study of translation edit rate with targeted human annotation, In *Proceedings of association for machine translation in the Americas*, 2006.

[31] Sutskever, I.; Vinyals, O.; Le, Q.V. (2014). Sequence to sequence learning with neural networks, *Advances in neural information processing systems*, 3104-3112, 2014.

[32] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. (2017). Attention is all you need, *arXiv preprint arXiv:1706.03762*, 2017.

[33] Wang, Q.; Li, B.; Xiao, T.; Zhu, J.; Li, C.; Wong, D.F.; Chao, L.S. (2019). Learning deep transformer models for machine translation, *arXiv preprint arXiv:1906.01787*, 2019.

[34] Wu, L.; Tian, F.; Qin, T.; Lai, J.; Liu, T.Y. (2018). A study of reinforcement learning for neural machine translation, *arXiv preprint arXiv:1808.08866*, 2018.

[35] Yang, B.; Li, J.; Wong, D.F.; Chao, L.S.; Wang, X.; Tu, Z. (2019). Context-aware self-attention networks, In *Proceedings of the AAAI Conference on Artificial Intelligence*, 33: 387-394, 2019.

[36] [Online]. Available: https://github.com/moses-smt/mosesdecoder, Accesed on 30 December 2017.

[37] [Online]. Available: https://github.com/Maluuba/nlg-eval, Accesed on 23 November 2019.

[38] [Online]. Available: https://pypi.org/project/pyter, Accesed on 7 Decemebr 2012.

[39] [Online]. Available: https://github.com/google/sentencepiece, Accesed on 10 January 2021.

[40] [Online]. Available: https://github.com/lovit/WordPieceModel, Accesed on 5 November 2018.

Cite this paper as:

D.X. Li, Z.Y. Luo. (2021). Regression Loss in Transformer-based Supervised Neural Machine Translation, *International Journal of Computers Communications & Control*, 16(4), 4217, 2021.
https://doi.org/10.15837/ijccc.2021.4.4217