Enhancing English-Persian Neural Machine Translation with a Large-Scale Parallel Dataset and Relative Position Representations

Alireza Kamyab, Negar Baghaei Nejad and Alireza Akhavanpour

December 9, 2024

Enhancing English-Persian Neural Machine Translation with a Large-Scale Parallel Dataset and Relative Position Representations

1st Alireza Kamyab
dept. of Computer Engineering
Shahid Rajaee University
Tehran, Iran
alirezakamyab19@gmail.com

2nd Negar Baghaei Nejad
dept. of Computer Engineering
Shahid Rajaee University
Tehran, Iran
negarbnejad@gmail.com

3rd Alireza Akhavanpour
Shenasa AI
Tehran, Iran
akhavan@shenasa.ai

***Abstract*—Transformer-based models have revolutionized neural machine translation (NMT), particularly with the introduction of the encoder-decoder architecture. However, training these models effectively often requires large amounts of parallel data or pretraining on massive unlabeled corpora. In the context of English-Persian translation, the lack of extensive parallel datasets has hindered progress. To address this, we introduce a new dataset of 4 million English-Persian parallel sentences that span various topics. Without any pretraining on unlabeled data, our model achieves a BLEU score of 47 on the PEPC benchmark and 35 on the MIZAN benchmark, demonstrating strong performance. We used Transformers with relative position representations, enabling the model to generalize to sequence lengths not seen during training. To promote further research and reproducibility, we have open-sourced both the dataset[1] and the trained model[2], supporting advancements in English-Persian NMT.**

***Index Terms*—neural machine translation, transformer, BLEU score, English-Persian dataset, attention mechanism**

I. Introduction

Neural Machine Translation is a machine translation approach which relies on neural networks to model the translations from source to target language. Other than the traditional statistical methods that are phrase-based, NMT uses only a single architecture, usually an encoder-decoder framework with an attention mechanism. This architecture enables the NMT model to understand deep patterns and dependencies among sequences, resulting in fluent and contextually correct translations. NMT has gained a variety of improvements, majorly brought about by the new Transformer-based models, such that the translation quality has increased tremendously and it has taken over the dominant approach in research and industry.

A. Recurrent Neural Networks

The RNNs have been widely used in machine translation tasks such as LSTMs [7] and GRUs. Given an input sequence $x = (x_1, \ldots, x_n)$, at any time step t, these models compute an embedding h_t using the previously computed hidden state h_{t-1}. This recurrent structure has some very useful properties for sequence modeling tasks, such as machine translation, where the order of the tokens is informative to the linguistic meaning. Besides, RNNs are often applied in time-series analysis, where the order of the values depends on temporal progressions.

However, RNNs do have some challenges when long sequences are involved since they have a tendency to "forget" the earlier embeddings as the sequence gets longer. While LSTM and GRU alleviate this problem by using various types of gating mechanisms, they still suffer in tracking information across very large sequences. In neural machine translation, it's based on the Encoder-Decoder architecture, where one critical task of an encoder is to encode an input sequence $x = \{x_1, \ldots, x_{T_x}\}$ into a vector $v \in \mathbb{R}^d$, with a fixed size that does not depend on whether the input is one word or a whole text. This could limit the model's capability for capturing long-range dependencies effectively, which is a critical requirement in translation.

B. Attention

Attention was first introduced in the decoder [2] to selectively concentrate on the relevant aspects of the outputs generated so far by the encoder in predicting the target sequence. In fact, attention enables learning of long-range dependencies without regard to their positional distance in the input sequence by allowing a model to dynamically attend to different states of the encoder. This mechanism significantly enhances the model's ability to capture more information about context; it is therefore especially effective in tasks such as machine translation.

C. Transformer Architecture

Although RNNs with attention mechanisms can handle longer text sequences, they still face inefficiency at very long sequences because of their sequential nature. This is because each state h_t depend on previous state h_{t-1}, and

[1]The dataset can be accessed at: https://github.com/AlirezaKamyab/English-Persian-NMT

[2]The model and code are publicly available at: https://github.com/AlirezaKamyab/NMT-Project

thus RNNs cannot be parallelized. The limitation overcome by the Transformer architecture [17] lays in the avoidance of recurrence and dependence on mere attention mechanisms, such as self-attention and multihead attention. This turned out to be an innovation which enabled the Transformers for the first time to outperform the state-of-the-art in machine translation.

For such a purpose, self-attention will enable this architecture to contextualize each token embedding in the context of a sentence where it may happen to be. This enhances long-range dependencies inside the representation. Another very important property of the Transformer architecture is filling in for the lack of recurrence with positional encoding. Positional encodings added to token embeddings will allow this model to capture word order successfully, hence being capable of distinguishing between the different word ordering.

D. Contributions

Here, we utilize an improved version of the Transformer that includes relative position encoding, enhancing its adaptability for sequences of variable lengths. For the difficult Persian WIKI PEPC [8] benchmark, our proposed model achieved a very high BLEU [12] score of 47, compared to previously reported results. On MIZAN [9] benchmark, we achieved BLEU score of 35.

We also introduce a new dataset with $4M$ parallel English-Persian sentence pairs, which has shown to significantly enhance the performance of our model. This dataset not only increases the diversity of training data but also improves the overall quality of translation results.

The remainder of this paper is organized as follows:

- **Section II** provides an overview of related work, highlighting the current advancements in the field and prior methodologies. This section begins with a discussion of the tokenizer employed in our experiments, followed by an explanation of a transformer variant utilized in the study.
- **Section III** outlines the methodology employed in this study, including the methods used for dataset collection, the training procedures, and a description of the hardware utilized.
- **Section IV** describes the experimental setup, including datasets, metrics, and evaluation criteria.

II. Related Works

A. Byte Pair Encoding

Subword tokenization techniques, such as Byte Pair Encoding (BPE), which [15] performed, have been applied in most of the NLP tasks to handle the occurrence of a rare or out-of-vocabulary word by subword units of that word that are more frequent. BPE was initially developed as a data compression algorithm that works by iteratively merging the most frequent pairs of symbols-character or character sequences-in a text corpus while building up a vocabulary of subwords which can efficiently represent even rare words. This is especially helpful in languages with rich morphology, where one word can have a large number of inflections.

BPE mitigates the problem of large vocabulary sizes and reduces the resulting requirement for high-dimensional word embeddings, respectively. It does this by breaking rare words into smaller subunits that appear more frequently. Consider a word like ”unhappily”; it could be split into subwords ”un”, ”happy”, and ”ly” so that it can be represented by the model without having to create a separate embedding. This further allows for better generalization because the model is able to deal with unseen words at inference time by breaking them down into known subwords.

While BPE has seen adoption in many transformer-based models, such as GPT [13] and BERT [4], it strikes a very effective balance between vocabulary size and expressiveness necessary for language modeling. Variants and improvements to BPE include SentencePiece [11], which further refined this approach to be more flexible for a variety of tasks entailing different languages and writing systems.

B. Transformer

The Transformer model, introduced by [17], revolutionized sequence-to-sequence tasks with an encoder-decoder architecture which relies on self-attention mechanisms to process input data in parallel rather than sequentially, increasing training efficiency significantly. The encoder consists of self-attention layers and position-wise feed-forward networks, while the decoder adds an encoder-decoder attention mechanism that enables it to pay attention to particular parts of the input sequence when decoding.

One of the most salient features of the Transformer model is its use of position encoding. While in RNNs, because of the nature of the feedforward processing step, the order in a sequence is inherently preserved, Transformers need to be explicitly informed about position to understand any order of tokens. [17] suggested a positional encoding based on the sinus function; These encodings are added to the input embeddings at the base of the encoder and decoder stacks. The positional encodings are designed to have the same dimensionality as the input embeddings, enabling a straightforward summation.

The encodings are derived using sinusoidal functions of varying frequencies, as defined by:

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{model}}}) \tag{1}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{model}}}) \tag{2}$$

where pos represents the token position and i indicates the dimension. The choice of sinusoidal functions facilitates the model’s ability to generalize to sequence positions not observed during training, as the positional encodings for any fixed positional offset k can be expressed as a linear function of the encoding at pos. This approach ensures that the positional information is smoothly integrated into the embedding space.

These generalize well to sequences of different lengths because positional information is given in such a way that it is independent from the length of the whole sequence. Later works further extend this with learned positional encodings and relative positional representations [16] [3], enhancing the model's capability to handle variable lengths of sequences not seen during training in a more robust manner.

Residual connections [6] and layer normalization [1] applied around each sub-layer proved to be crucial in practice for reducing training instability as well as improving gradient flow. These architectural modifications not only make possible the efficient learning but propagate the positional information through the depth, a necessity in modeling sequential dependencies.

1) Self-Attention: Self-attention leverages a dot-product compatibility function for computing relationships between elements in a sequence. Introduced by [17], self-attention employs multi-head attention, allowing attention heads to operate in parallel across sequence positions for efficiency.

Given an input sequence $x = (x_1, \ldots, x_n) \in \mathbb{R}^{d_x}$, each attention head computes an output sequence $z = (z_1, \ldots, z_n) \in \mathbb{R}^{d_z}$. In this work, we adopt relative position representation, as described by [16], instead of absolute position representation. This approach introduces vectors a_{ij}^V and a_{ij}^K, which capture the positional relationship between elements x_i and x_j. The output of multi-head attention is computed as a weighted sum of linearly transformed inputs, adjusted by relative position information:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V), \tag{3}$$

where each weight coefficient α_{ij} is derived using a softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}. \tag{4}$$

Each score e_{ij} is computed via scaled dot-product:

$$e_{ij} = \frac{x_i W^Q (x_j W^K)^T + x_i W^Q (a_{ij}^K)^T}{\sqrt{d_z}}. \tag{5}$$

The computations in equations (1) and (3) can be efficiently parallelized.

C. Relative Position Information Representation

Relative position representation was first proposed by [16] as an alternative to absolute position encoding. In the context of linear sequences, relative position embeddings capture the positional relationships between tokens without relying on specific sequence indices.

In natural language processing, each token can be represented as a vertex in a directed graph. The relative embedding would represent its position relative to another token. The model achieves this without keeping track of the absolute sequence positions but rather the relative distance and direction of one token to another. For instance, whether it is on the left or right side. This way, the model generalizes well on the different lengths of sequences than what it has seen during training.

It is assumed that the exact relative positioning has less significance beyond a certain distance; hence, more distant relationships will contribute less to token-level dependencies. To enhance generalization to sequence lengths not found during training, we learn a set of $2k + 1$ relative position embeddings. These embeddings correspond to distances from $-k$ to k, where $k < 0$ represents positions to the left of the reference token, $k > 0$ represents positions to its right and $k = 0$ denotes a self-looping edge on the token itself.

$$a_{ij}^K = w_{\text{clip(j - i, k)}}^K \tag{6}$$

$$a_{ij}^V = w_{\text{clip(j - i, k)}}^V \tag{7}$$

Then $w^K = (w_{-k}^K, \ldots, w_k^K)$ and $w^V = (w_{-k}^V, \ldots, w_k^V)$ are learned.

III. Our Work

A. Proposed Architecture

Our proposed model is a Transformer model-based architecture with an encoder and decoder structure, each having $N = 6$ layers stacked. Each encoder layer shall have a self-attention sublayer followed by a position-wise feed-forward network. The decoder layers are similar to the standard transformer decoders and contain three sublayers: a self-attention sublayer, a cross-attention sublayer (attending to the encoder outputs), and a position-wise feed-forward sublayer. Residual connections and layer normalization after each sublayer further help train stabilization and propagate positional information through higher layers.

To ensure that the output of decoders is generated sequentially, in each layer of the decoder, we modify the self-attention mechanism such that attending to subsequent positions is not possible. Thus, this masks the attention allowing only previous and current positions and ultimately preserves autoregressive behavior. It allows the model to predict each token based solely on prior context.

We use a Byte Pair Encoding tokenizer that splits each input word into subword units. Further, these subwords are embedded into the vector space $\mathbb{R}^{\text{d\_model}}$, providing input embeddings both for the encoder and decoder.

Our model is trained on approximately 4 million parallel English-Persian sentence pairs with no extra pretraining on unlabeled data. To better generalize to sequence lengths unseen during training, we take $k = 16$ where k defines the clip value for the maximum distance in relative position embeddings, for the model's sensitivity to longer-range dependencies diminishes.

B. Dataset

Due to the rarity of an English-Persian parallel dataset, a corpus of approximately 4 million parallel sentence pairs in English and Persian was gathered from movie subtitles, books, news, and commonly used idioms. Also, a portion of the sentences in English were retrieved from the WMT19 dataset [5] and translated into Persian. The normalization process included removal of accent symbols on some of the English sentences and removal of duplicate entries. Sentences were then manually checked for linguistic correctness. For Persian, normalization was done on all sentences so that they are in uniform form. In order to avoid cross-linguistic interference, sentences containing words of languages other than English or Persian were excluded from this dataset. The collected dataset contains a wide range of sentence types so that the model can get a strong foundation for different usage of languages. This varies from formal language and mathematical and scientific expressions to everyday conversation phrases, making it versatile in capturing a wide range of linguistic styles. This includes formal sentences that help to equip the model with how to handle official or structured language mainly found in professional or academic contexts. The mathematical and scientific sentences introduce technical terminologies and specialized structures that are necessary for accurate translations in these domains. The conversational phrases enrich the dataset with casual and idiomatic language that increases the versatility of the model in real-world, informal conversations. This kind of diverse composition will definitely enable the model to generalize better across contexts and produce translations that are contextually appropriate and nuanced.

C. Training

1) Objective: The decoder is trained to predict the next probable token y_t given the previous tokens $y_{0:t-1}$, input sequence x, and model parameters θ. The probability of predicting token y_t at each time step t is defined as:

$$p(y_t) = \prod_{t=1}^{T} p\left(y_t \mid y_{0:t-1}; x; \theta\right) \tag{8}$$

To optimize the model, we use a maximum likelihood objective, defined as:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log p\left(y_t \mid y_{0:t-1}; x; \theta\right) \tag{9}$$

This objective maximizes the likelihood of the target sequence by minimizing the negative log probability of the correct tokens at each time step.

2) Hardware: The experiments were performed with a single NVIDIA RTX 3090 GPU. Given the choice of hyperparameters described in the *Experiments* section, our base model needed 30 hours of training to complete a total of $796,200$ steps whereas bigger model was trained for 48 hours over $796,200$ steps.

3) Optimizer and Scheduler: We employed the Adam optimizer [10] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. The learning rate was controlled using a warmup scheduler, following the approach proposed by [17]:

$$\text{lrate} = d_{\text{model}}^{-0.5} \min\left(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}^{-1.5}\right) \tag{10}$$

The learning rate is increased linearly during the initial $warmup$ training steps and subsequently decays proportionally to the inverse square root of the training steps. During the warmup phase, the learning rate starts at a very low value to prevent divergence, as the weights are initially randomly initialized.

4) Regularization: **Dropout** was also applied on the output of every sublayer before residual addition and normalization. The dropout rate throughout all the experiments is set to 0.1. Besides, **label smoothing** is used during training with a smoothing factor of 0.1 to let the model be less confident during training, which improves generalization.

IV. Experiments

The proposed model was evaluated on two datasets: the PEPC dataset [8], a parallel English-to-Persian dataset derived from Wikipedia, and the MIZAN dataset [9], a Persian-English parallel corpus containing over one million sentence pairs from literary masterpieces. The evaluation focused on translating English sentences into Persian, with beam search omitted from all experiments. Performance was measured using the BLEU score [12], computed with the NLTK library.

Given that the original study does not report baseline results for the PEPC [8] and MIZAN [9] datasets, and no other validations or comparisons of this methodology on these datasets could be found, we have gone ahead to compare the performance of our model with regard to these datasets. While it is difficult to make direct comparisons without specific baseline numbers, this approach is still useful in judging the relative effectiveness of our method. In order to be transparent, we henceforth acknowledge this limitation and stress that our results are based on the available datasets. Furthermore, we provide detailed descriptions of our evaluation setup, including dataset versions, pre-processing steps, and evaluation metrics, to allow for reproducibility and validation. We also release our trained models and code publicly so that others can replicate our experiments and verify our results. More importantly, rather than relying solely on quantitative measures, we focus on the insight such an analysis brings into a better understanding of the task and how effective our approach is. We also considered other baselines from relevant studies to expand the contextual scope of our comparison.

A. Evaluation Dataset Description

The PEPC dataset [8] comprises 200,000 parallel English-Persian sentence pairs with varying sequence lengths. The average sequence length is 61 tokens, while the maximum sequence length reaches 153 tokens.

The MIZAN dataset is a Large Persian-English Parallel Corpus by [9], is a comprehensive resource for Persian-English translation tasks. This parallel corpus consists of over one million sentence pairs, primarily derived from Persian translations of English literary works. The dataset's focus on literary content ensures a rich variety of linguistic features, including complex sentence structures, diverse vocabulary.

B. Results

We have conducted experiments with two variants of the Transformer model in order to inspect to what extent position representation affects translation quality. The first model uses relative position representation and hence has the ability to capture contextual relations without relying on absolute position indices. In contrast, the second model uses an absolute position representation which allocates fixed positional embeddings to every token with respect to its position in a sequence. This comparative setup provides the opportunity to analyze how each of the approaches influences the model's translation performance.

TABLE I
BLEU METRIC COMPARISON OF TRANSFORMER MODELS WITH ABSOLUTE AND RELATIVE POSITION REPRESENTATIONS. RESULTS ARE PRESENTED FOR BOTH SMALL (S) AND LARGE (L) VARIANTS OF EACH MODEL TYPE.

Benchmark	**Absolute POS**		**Relative POS**	
	Model S	***Model L***	***Model S***	***Model L***
PEPC	43	43	45	**47**
MIZAN	31	32	33	**35**

According to the tabel I even the small transformer configuration with relative-position representation achieves a higher BLEU score than the absolute variant. Although [17] suggests that the model generalizes across all sequence lengths, the relative-position method [16] appears to enable generalization to sequence lengths not encountered during training. Furthermore, Fig. 1 illustrates that the accuracy of the absolute-position model declines as sequence length increases, whereas the accuracy of the relative-position model remains consistent across varying sequence lengths.

Table I demonstrates that the model's performance is relatively lower compared to the PEPC dataset. This discrepancy can be attributed to the MIZAN dataset containing sentences derived from literature, which pose greater challenges for translation. Nonetheless, the model exhibits strong capabilities in accurately translating idiomatic expressions.

NEXT STEPS

We demonstrated that a straightforward Transformer architecture that incorporates relative position representations achieves a BLEU score of 47 on PEPC and 35 on MIZAN just by training on the proposed dataset. Recent advances in sequence transduction frequently employ transformer models only with a decoder [13]. These models are typically pretrained on unlabeled data using objectives such as next-token prediction or denoising. Alternatively, models like T5 [14] maintain the encoder-decoder framework but reframe the task within a text-to-text paradigm, pretraining on unlabeled data before fine-tuning on specific downstream tasks.

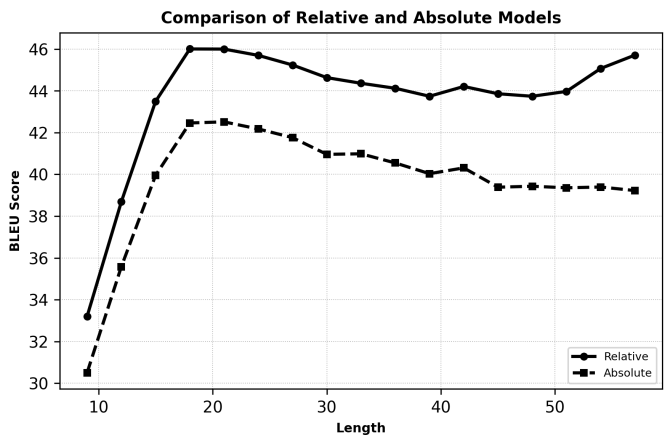

Fig. 1. This figure illustrates a comparison of BLEU scores between two Transformer variations utilizing absolute and relative position representations. The plotted data includes a smoothing factor of 0.8.

Given the abundance of unlabeled data, we hypothesize that employing a similar pretraining strategy, followed by fine-tuning on our dataset, has the potential to achieve state-of-the-art performance.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML]. URL: https://arxiv.org/abs/1607.06450.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL]. URL: https://arxiv.org/abs/1409.0473.

[3] Zihang Dai et al. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. 2019. arXiv: 1901.02860 [cs.LG]. URL: https://arxiv.org/abs/1901.02860.

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[5] Wikimedia Foundation. *ACL 2019 Fourth Conference on Machine Translation (WMT19), Shared Task: Machine Translation of News*. URL: http://www.statmt.org/wmt19/translation-task.html.

[6] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[8] Akbar Karimi, Ebrahim Ansari, and Bahram Sadeghi Bigham. "Extracting an English-Persian parallel corpus from comparable corpora". In: *arXiv preprint arXiv:1711.00681* (2017).

[9] Omid Kashefi. *MIZAN: A Large Persian-English Parallel Corpus*. 2020. arXiv: 1801.02107 [cs.CL]. URL: https://arxiv.org/abs/1801.02107.

[10] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[11] Taku Kudo and John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. 2018. arXiv: 1808.06226 [cs.CL]. URL: https://arxiv.org/abs/1808.06226.

[12] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Ed. by Pierre Isabelle, Eugene Charniak, and Dekang Lin. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[13] Alec Radford. "Improving language understanding by generative pre-training". In: (2018).

[14] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG]. URL: https://arxiv.org/abs/1910.10683.

[15] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. 2016. arXiv: 1508.07909 [cs.CL]. URL: https://arxiv.org/abs/1508.07909.

[16] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. *Self-Attention with Relative Position Representations*. 2018. arXiv: 1803.02155 [cs.CL]. URL: https://arxiv.org/abs/1803.02155.

[17] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.