POS-Tagging based Neural Machine Translation System for European Languages using Transformers

[1]PREETHAM GANESH, [2]BHARAT S. RAWAL, [3]ALEXANDER PETER, [4]ANDI GIRI

[1]University of Texas at Arlington, Arlington, TX, USA
[2]Gannon University, Erie, PA, USA
[3]Softsquare, Silver Spring, MD, USA
[4]Softsquare, Toronto, ON, CANADA

Abstract: The interaction between human beings has always faced different kinds of difficulties. One of those difficulties is the language barrier. It would be a tedious task for someone to learn all the syllables in a new language in a short period and converse with a native speaker without grammatical errors. Moreover, having a language translator at all times would be intrusive and expensive. We propose a novel approach to Neural Machine Translation (NMT) system using inter- language word similaritybased model training and Part-Of-Speech (POS) Tagging based model testing. We compare these approaches using two classical architectures: Luong Attention-based Sequence-to-Sequence architecture and Transformer based model. The sentences for the Luong Attention-based Sequence-to-Sequence were tokenized using SentencePiece tokenizer. The sentences for the Transformer model were tokenized using Subword Text Encoder. Three European languages were selected for modeling, namely, Spanish, French, and German. The datasets were downloaded from multiple sources such as Europarl Corpus, Paracrawl Corpus, and Tatoeba Project Corpus. Sparse Categorical CrossEntropy was the evaluation metric during the training stage, and during the testing stage, the Bilingual Evaluation Understudy (BLEU) Score, Precision Score, and Metric for Evaluation of Translation with Explicit Ordering (METEOR) score were the evaluation metrics.

Key Words: Neural Machine Translation, Luong Attention, Self-Attention, Transformer.

Received: November 3, 2020. Revised: May 8, 2021. Accepted: May 17, 2021. Published: May 31, 2021.

1. Introduction

The total number of languages spoken by human beings worldwide is approximately equal to 6500 [1], where the total number of countries in the world is 195. The total population in the world is approximately 7.82 billion [2]. Based on a report by Ethnologue: Languages of the World [3], English is spoken by 1.27 billion people (16.22%), 537.9 million (6.88%) people speak Spanish, 276.6 million people speak French, 131.6 million (1.8%) people speak German, and 67.7 million (0.9%) people speak Italian. On the other hand, English is the official language in 93 countries, French is the official language in 29 countries, Spanish is the official language in 21 countries, German is the official language in 9 countries, and Italian is the official language in 4 countries [4]. An official language is also called a state language, i.e., a language given an essential status in a particular country, state, or other jurisdiction [5]. Even though many people speak the above mentioned European languages, the count of people who can read/write/speak two languages simultaneously is meager. Since many countries have only one official language for documentation of government decisions, text-based language translation can solve this problem and benefit many people.

According to a report [6], it is tough to learn a new language like a native speaker at an old age. In this modern deep learning era, the text can be translated from one language to another with a trained neural network, which will be easier to use than learning a new language or having a professional language translator. It can help in translating official text released by the Central Government and various State governments. It can help communicate between two people from two different ethnic backgrounds or two different places where their official language differs. A solution to this would be to use a professional language translator. However, a professional language translator will face three problems: number of sentences translated per day, number of words in memory, and translation quality.

A professional language-translator can only translate up to 2000 or 3000 words per day or a few hundred sentences per day, whereas a trained Machine Translation Neural Network model can translate large amounts of text in microseconds. The vocabulary of words a professional may know might be limited. However, if a computing device with a large enough memory is available, then the trained Machine Translation neural network model might hold a large enough vocabulary for translating sentences. Also, a trained Machine Translation neural network model will be able to translate the given text with excellent quality when compared to the professional [7]. Hence, it is highly essential to translate text from one language to another, and for that, the Machine Translation model is less intrusive, less expensive, and more efficient than a professional language translator.

Even though there are good research works published regularly on Machine Translation, the solutions proposed by many researchers can always be improved to provide better solutions with different approaches and optimization techniques. However, there are many problems faced by researchers who are working on improving the current state-of-the-art Machine Translation models. Some problems faced by the researchers when trying to improve the current state-of-the-art models for European Languages are as follows: (1) Out-Of-Vocabulary words, (2) less number of sentences in a data, (3) Rare words in a language, (4) long sentences, (5) ability to control the quality of translated output [8]. The work done in this paper

is an attempt to solve some of the problems mentioned above.

The structure of the work presented in the paper is as follows: Section II describes the related work on Machine Translation models. Section III explains the methodology and process flow for the proposed solution. Section IV discusses in detail about the derived results. Section V concludes the paper based on the results derived.

2. Related Work

This section describes in detail the work done by authors in text-based language translation using Statistical and Neural Network approaches. It also discusses how authors used different approaches from language context to improve the Machine Translation models' efficiency and precision.

There two types of Machine Translation, namely (1) Statistical Machine Translation and (2) Neural Machine Translation. Statistical Machine Translation (SMT) is the process of translating text using on statistical models where the parameters are derived from parallel text corpora [9]. Neural machine translation (NMT) is the process of translating text using neural network models to predict the likelihood of sequence of words. NMT approach typically models entire sentence within a single integrated model [10]. When a standard NMT model is combined with different attention mechanisms and subword-nmt methods, it produces impressive results.

One of the first NMT models was designed by the researchers at Google, commonly called the Sequence-to-Sequence model [11]. It has multiple uses such as response generation, language translation, image captioning, and text summarization [12]. The Sequence-to-Sequence (Seq2Seq) model consists of 2 sub-modules: An Encoder model and a Decoder model. The Encoder takes input from one language and encodes it with the help of Recurrent Neural Networks (RNN). The results are passed into the decoder, which then decodes it to the output language with the help of RNN. Bahdanau et al. in [13] introduced the concept of Attention to the Seq2Seq architecture, as the normal Seq2Seq failed to work for longer sentences (more than 40 words or tokens). The authors' idea presented in the paper was to use the input and the previous timestep's output to predict the output of the current timestep. The authors proved that this produced better results than the previous Seq2Seq model [11] for longer sentences.

Luong et al. in [14] provided different approach to the Attention-based Seq2Seq model for NMT. The Bahdanau Attention model is considered a Local Attention model, whereas the Luong Attention model is a Global Attention model. The Luong Attention model aims to take all the encoder's hidden states to derive the context vector. The difference between the Luong Attention model and the Bahdanau Attention model is that the Bahdanau Attention model takes the output of the previous timestep from the decoder and the top layer's output from the encoder. However, the Luong Attention model takes the output of the topmost Long Short-Term Memory (LSTM) layer of both the Encoder and Decoder models for calculating the context vector.

One of the problems faced by researchers working on improving NMT models is Out-Of-Vocabulary (OOV) words. Sennrich et al. provided a solution to that in [15] called Byte-Pair-Encoding (BPE). The idea proposed was to use a data compression technique that would replace a rare word with the most frequent pair of bytes. It helps in the prefix, suffix separation, and composite splitting of a rare word in a language. It also reduces the memory used for training the RNN model by converting all the words in the training data into sub-word units. An improvement to this was provided by Google's researchers Kudo et al. in [16], where they introduced another type of sub-word unit algorithm called SentencePiece, which supports BPE and Unigram-Language-Model. The SentencePiece model processes sentences with a lesser memory footprint compared to the original BPE model.

Wu et al. in [17] using the SentencePiece model, where the authors tried a stacked LSTM Encoder-Decoder model with Beam search, which produced better results than the results in the previous works done by researchers. Since the Bi-directional LSTM based Encoder-Decoder model with complex architecture took longer training time and inference time, the researchers at Facebook Gehring et al. in [18] proposed a Convolutional Encoder-Decoder model which produced the similar results to work in the [17] with an inference method that is twice as faster with higher accuracy. An improvement to this was produced by the Google researchers Vaswani et al. in [19]. The authors proposed a Self-Attention based Dense Neural Network (DNN) NMT model, commonly called as Transformer architecture, which computes the context vectors just based on Multi-layer DNN where each layer DNN layer is attached to a Self-Attention layer.

The Self-Attention layer allows the input to interact to identify the words that need more attention. There are two advantages of Self-Attention over other Attention architectures: (1) Capability to perform parallel computing (in comparison with RNNs based Encoder and Decoder models); (2) Lesser need for Deep RNN architectures, which take more time compared to Deep DNN architectures. It helps in a more effortless flow of gradients through all the states, which helps solve the vanishing gradient problem to some extent. The Multi-head attention helps the model cooperatively attend the statistics from unique representation subspaces at unique positions.

Another problem faced by a researcher with the NMT models is that it translates every word it receives. That would not work for words in the Noun category, specifically the Proper Noun categories. Authors provided a solution to this in [20], [21], where the authors tried to use Parts-of-Speech (POS) Tagging-based approach to overcome the problem mentioned above. In [20], the authors proposed an NMT model with a tag prediction attention mechanism. For this purpose, NMT and POS-Tagging were jointly modeled using Multi-Task learning, where coarse attention mechanism was used on input annotation and target hidden states to predict the target annotation and the target word.

In this paper, we propose a novel approach to the NMT by

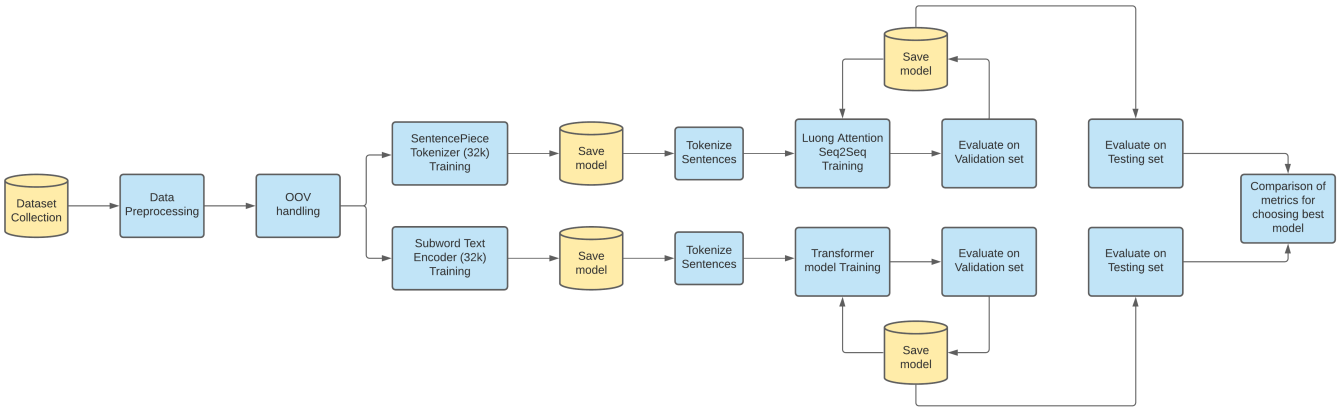

Fig. 1. Process Flow

using inter-language word similarity-based model training and POS-Tagging based model testing. This approach is modeled on two classical NMT architectures: Luong Attention-based Sequence-to-Sequence architecture and Transformer based model. The datasets used for developing NMT models were obtained from modeling the popular datasets such as Europarl, Paracrawl which are mentioned in Section III-A. The sentences in the dataset were tokenized into subword units using SentencePiece Tokenizer and Subword Text Encoder tokenizer, which is mentioned in detail in Section III-D. The OOV handling of the sentences in the dataset is explained in detail in Section III-C. The trained models were evaluated using the metrics mentioned in detail in Section III-F, where the analysis was based on how well the model performed on the validation and testing sets as the length of input sentence changes.

3. Process Flow

This section describes in detail the datasets, used for each of the NMT models, and also explains the various concepts used in the experimental setup. Fig. 1 shows the process flow for the proposed architecture.

3.1. Dataset Description

Multiple parallel corpora have been used for training the model such as Tatoeba project [22], Europarl corpus [23], and Paracrawl [24]. The details of the individual datasets and the combined datasets before the pre-processing step is given in Table I.

TABLE I
DETAILS OF INDIVIDUAL AND COMBINED DATASETS BEFORE PRE-PROCESSING

Translation Task	Dataset	No. of Sentences	
		Individual	Combined
English-French	Europarl	2.007M	33.567M
	Paracrawl	31.374M	
	Tatoeba	0.186M	
English-German	Europarl	1.92M	18.411M
	Paracrawl	16.264M	
	Tatoeba	0.227M	
English-Spanish	Europarl	1.965M	24.081M
	Paracrawl	21.987M	
	Tatoeba	0.128M	

3.2. Data Pre-processing

The NMT system was built with an AMD Ryzen 5 series 6 core CPU and an NVIDIA 2080 super GPU. All the datasets collected from the work done by the authors in [23], [24] contained a lot of errors/noise such as Extended Markup Language (XML) tags, unwanted characters (characters other than a-z, 0-9, and a few punctuation symbols), empty sentences, and many more. There were also many duplicate sentences and multiple translations for the same sentence. The problems mentioned above might confuse the NMT models in choosing the right word in a sequence of words predicted by the model. The first step of data pre-processing was to split the combined dataset into smaller datasets where each dataset contained 1M pair of sentences. The process done in the first step was to make use of the multi-threading capability to speed up the data pre-processing. The next step of data pre-processing was to lower case sentences, remove the unwanted noise characters, and remove duplicates. To restrict Graphics Processing Unit (GPU) memory usage, sentences containing more than 40 tokens or words were dropped from the dataset. It took approximately 0.001 seconds to process one pair of sentences from each language, where the number of threads used to pre-process the datasets concurrently is 6. The details of the datasets after pre-processing are given in Table II.

3.3. Out-of-vocabulary Handling

As mentioned in Section II, any word sent into an NMT model is translated into the target language work, but the problem arises when a Proper Noun or a rare word is passed. It is because the pronunciation and spelling of a proper noun are the same in all languages. A solution to this was to convert the proper nouns mentioned above into a unique token such as '<unk>.' However, two problems arise, which are related to the count of rare words or Proper Noun in the testing input sentence. Problem 1 is the allocation of different unknown

TABLE II
DETAILS OF THE DATASET AFTER PRE-PROCESSING

Translation Task	Dataset	No. of sentences
English-French	Train	24.248M
	Validation	4852
	Test	4852
English-German	Train	16.022M
	Validation	5612
	Test	5612
English-Spanish	Train	19.105M
	Validation	5257
	Test	5257

tokens (such as '<unk1>', '<unk2>', ..) for more than one rare word in a sentence, as there would be no limit if the training set is large. Problem 2 is the order in which these sequence unknown tokens would be predicted by the model if all the rare words are just identified with a single unknown token because the order of words may differ from one language to another.

A solution to these two problems mentioned above would be to convert the rare words or Proper Nouns into a single format based unknown token. For example: the word 'preetham' is converted to '<p#r#e#e#t#h#a#m>'. However, the problem now was identifying the proper nouns in each sentence from the training set. One of the solutions would be to use special packages such as Natural Language ToolKit (NLTK) [25], or Spacy [26]. The problem with these packages was that they would atleast 0.01 seconds to process one sentence, and when the number of sentences is in millions, it will take days or weeks to process the dataset.

We used an inter-language word similarity-based approach to extract the unique tokens from the training set for this problem. First, extract the rare words in the corpus with just one occurrence in the corpus, and then choose the words with a length of 8 (because, when the sentences are tokenized using custom tokenizers such as SentencePiece tokenizer or Subword Text Encoder, where the length of such sentences might go beyond the threshold length, i.e., 40). The tokens extracted from the training set are used for converting these unique tokens in the validation and testing sets. During the testing stage, packages such as NLTK [25], or Spacy [26] are used to extract the Proper Noun or rare words from the sentences.

3.4. Sentence Tokenization

Two types of sentence tokenization were used on the OOV-handled datasets; namely, SentencePiece Tokenizer [16], and Subword Text Encoder [19]. SentencePiece tokenizer was used for the Luong Attention-based Seq2Seq model, and Subword Text Encoder was used for the Transformer model. Unigram algorithm was used for training the SentencePiece model, where the vocabulary size was approximately 32k. Similarly, for the Subword Text Encoder, the vocabulary size was approximately set as 32k, i.e., 2^{15}. Since there was a shortage of memory availability during the SentencePiece tokenizer training, 18M sentences were sampled from the new training set resulted from the Section III-C. However, this was not the problem with the Subword Text Encoder tokenizer. After these tokenizers were trained, the sentences from the training, validation, and testing sets were encoded. Since the tokenizer divides original words into subwords, the number of tokens in each sentence increased when split by space (' '). Hence, similar to Section III-B, sentences that have more than 40 tokens were dropped. The details of the dataset after tokenization of sentences are given in Table III.

TABLE III
DETAILS OF DATASETS AFTER SENTENCE TOKENIZATION

Translation Task	Model	No. of Sentences		
		Training	Validation	Test
English-French	SentencePiece	20.144M	4020	4051
	Subword Text	21.916M	4381	4386
English-German	SentencePiece	14.236M	4980	5006
	Subword Text	14.875M	5185	5216
English-Spanish	SentencePiece	16.673M	4548	4607
	Subword Text	17.470M	4782	4826

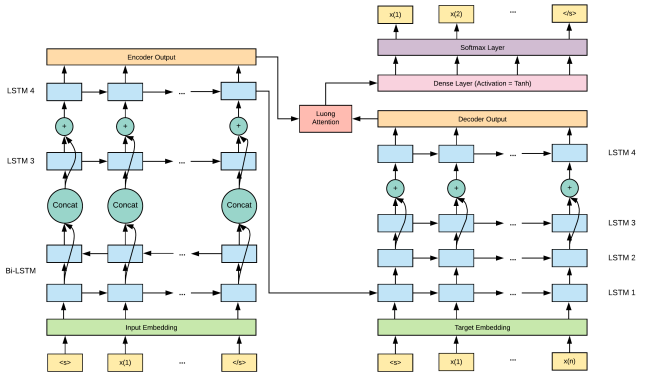

Fig. 2. Luong Attention-based Seq2Seq model

3.5. Neural Network Architectures

There are two main types of neural network architectures used for developing the NMT models for each language; namely, Seq2Seq architecture [11], and Transformer architecture [19]. The architecture diagram for the Seq2Seq model is given in Fig. 2. The Seq2Seq architecture consists of 2 models, the encoder model, and the decoder model. The encoder model consists of 5 layers, 1 Embedding layer, 1 Bi-directional LSTM layer (one forward LSTM layer and one backward LSTM layer), and 2 layers of unidirectional LSTM layers. In the bi-directional LSTM, the first layer traverses from left-to-right (one forward LSTM layer), while the next layer traverses from right-to-left (one backward LSTM layer). A bi-directional layer is chosen as the first layer in the encoder model because it helps obtain the context and other essential features [13]. The equations used for calculating the bi-directional layer's output and states are given in (1).

$$
\begin{aligned}
x_t^f, m_t^f, c_t^f &= LSTM_f(x_t^0, m_t^0, c_t^0; W^f) \\
x_t^b, m_t^b, c_t^b &= LSTM_b(x_t^0, m_t^0, c_t^0; W^b) \\
x_t^1 &= concat(x_t^f, x_t^b) \\
m_t^1 &= concat(m_t^f, m_t^b) \\
c_t^1 &= concat(c_t^f, c_t^b)
\end{aligned} \tag{1}
$$

In (1), $LSTM_f$ & $LSTM_b$ are the forward and backward LSTM layers, where its parameters are W^f, and W^b. x_t^0, m_t^0, and c_t^0 are the inputs to the forward and backward LSTM layers, where x_t^0 is input sequence, and m_t^0, and c_t^0 are the initial hidden states i.e. initial memory state and initial carry state at time step t. x_t^1, m_t^1, and c_t^1 are the concatenated outputs, which would be input to the next unidirectional LSTM layer.

It can be noticed from Fig. 2 that both the encoder model and the decoder model contain residual connections between the 2nd and 3rd LSTM layers. These connections are used because when more LSTM layers are stacked together, the model will suffer from vanishing gradient problems [27], [28]. In other words, the deeper the model, the more it forgets about the information it has seen. The equation used for calculating the residual output is given in (2). In the decoder model from Fig. 2, we use Luong Attention after the stacked LSTM layer, followed by two dense layers, one with a Tanh activation function and the other with a Softmax activation function.

$$
\begin{aligned}
x_t^i, m_t^i, c_t^i &= LSTM_i(x_t^{i-1}, m_t^{i-1}, c_t^{i-1}; W^i) \\
x_t^i &= x_t^i + x_t^{i-1} \\
m_t^i &= m_t^i + m_t^{i-1} \\
c_t^i &= c_t^i + c_t^{i-1} \\
x_t^{i+1}, m_t^{i+1}, c_t^{i+1} &= LSTM_{i+1}(x_t^i, m_t^i, c_t^i; W^{i+1})
\end{aligned} \tag{2}
$$

The architecture used for training the Transformer model is given in Fig. 3. Like the Seq2Seq architecture, the Transformer architecture consists of two models: an encoder model and a decoder model. These models have three sub-modules: positional encoding, multi-head attention, and a feed-forward network. The transformer model does not contain any recurrent network, because of which the model might not know the positioning of words in a sentence. The authors in [19], used positional encoding to encode the relative position between words in a sentence. Self-attention was used in the Transformer model, which mainly consists of 3 inputs, namely, Q (query), K (key), and V (value). The equation used for calculating attention output (per head) is given in (3).

$$
Attention(Q, K, V) = softmax_k(\frac{QK^T}{d_k^{1/2}})V \tag{3}
$$

The transformer model uses multi-head attention, where each head consists of the self-attention architecture. The encoder model consists of one multi-head attention module followed by a feed-forward network. The decoder model consists of two multi-head attention modules, followed by a feed-forward network. A dense layer follows the decoder

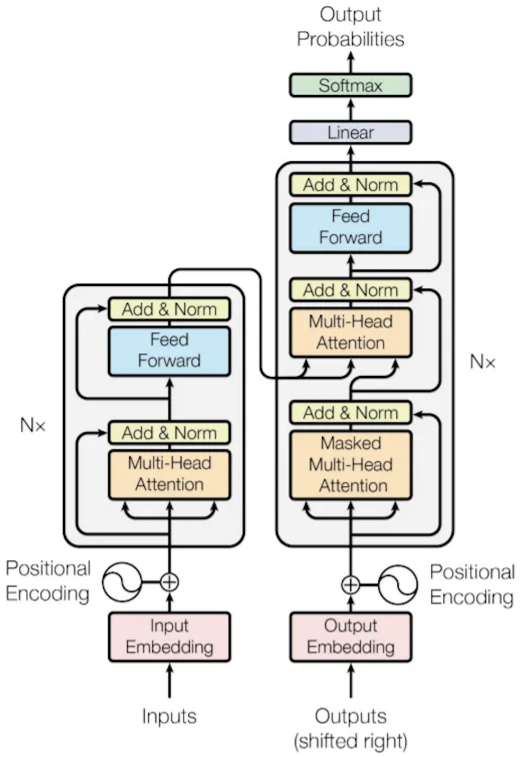

Fig. 3. Transformer model architecture from [19]

model with linear activation followed by a dense layer with a softmax activation.

3.6. Evaluationmetrics

1) Precision: Precision is the fraction of accurately estimated positive observations to the total estimated positive observations as given in (4).

$$
Precision = \frac{TP}{TP + FP} \tag{4}
$$

In (4), TP represents the number of true positives, and FP represents the number of false positives.

2) Brevity: It is a score used for evaluating the concise and exact use of words by the model in translating sentences.

3) Bilingual Evaluation Understudy (BLEU): It is a method used to measure the difference between machine translation and human translation [29]. It works on matching n-grams in the predicted translation to n-gram in the actual translation/reference text. The score ranges from 0.0 to 1.0 or 0% to 100% as given in (5).

$$
BLEU = Precision * Brevity \tag{5}
$$

4) Metric for Evaluation of Translation with Explicit Ordering (METEOR): It is a metric used for translating the machine translation output, where the calculation is based on the harmonic mean of unigram precision and recall [30]. The weight of recall is higher than precision.

4. Results and Discussions

This section discusses in detail the results obtained on performing the language translation task mentioned in Section III. The models trained and tested in this section were implemented with the help of TensorFlow [31].

4.1. Hyperparameter Selection for the Neural Networks

The neural network's hyperparameter values should be tuned based on the dataset in order for the neural network to train and predict efficiently and precisely. As mentioned in Section III-E, the Seq2Seq architecture consists of 2 models: an encoder and a decoder. The encoder consists of five layers: one embedding layer, one bi-directional LSTM layer, and two unidirectional LSTM layers. The decoder consists of eight layers: one embedding layer, four unidirectional LSTM layers, one Luong attention layer, and two dense layers. The number of units in all the layers except the topmost layer with softmax activation is 512, and the number of units in the topmost layer with softmax activation is the vocabulary size of the target language $\approx$ 32k. Adam optimizer [32] was used for training the model where the learning rate used was 0.001.

Similar to the Seq2Seq model, the transformer model consists of an encoder and a decoder. The encoder and decoder have three sub-modules: positional encoding, multi-head attention, and a feed-forward network. The encoder consists of 6 layers, where the number of units in each layer is 512, the number of units in the feed-forward layer is 2048, and the number of attention-heads is 8. Similar to the encoder, the decoder consists of 6 layers, where the number of units in each layer is 512, the number of units in the feed-forward layer is 2048, and the number of attention-heads is 8. The transformer model was trained using Adam optimizer [32], where the equation for learning rate is given (6) [19], beta_1 = 0.9, beta_2 = 0.98, and epsilon = 1e-9.

$$rate = d_{\text{model}}{}^{-0.5} * min(step\_num^{-0.5}, warmup\_step^{-1.5}) \quad (6)$$

4.2. Training and Testing of the Neural Network Models

Sparse Categorical CrossEntropy was the metric used during the training stage of both the Seq2Seq model and Transformer model. Both the models were trained for 200k training steps. Since the dataset size was large, i.e., 150k training steps per epoch, once in every 10k steps, the model is validated on the validation data. If the validation loss has decreased, the model's checkpoint would be saved, and the model would continue training. If the validation loss has not decreased, then the patience count would be incremented by one, and the model would continue training. If the patience count is greater or equal to 4, the model will stop training. The Seq2Seq model took 0.57 seconds to train on the NVIDIA GeForce 2080 Super graphics card, while the Transformer model took 0.46 seconds to train on the graphics card. Both the models had a batch size = 128.

The models' performance on the validation and testing sets are given in Table IV and Table V. The models' performance on the testing set based on length of input sentence using Precision Score and Brevity Score are given in Table VI. Similarly, the models' performance on the testing set based on length of input sentence using BLEU score and METEOR Score are given in Table VII.

4.3. Performance of the Models on the Validation and Testing Sets

It can be observed from Table IV and Table V that the performance of the Seq2Seq model and Transformer model are similar for provided language combinations. It can also be observed that for the provided language combinations, the Transformer model provides better performance than the Seq2Seq model. For the French-English language combination, it can be observed that the Precision Score is almost the same for both models. However, BLEU Score's change comes from the improved value of the Brevity Score for the Transformer model. The increase in Precision and Brevity Scores improved BLEU Score for the Transformer model compared to the Seq2Seq model for the other language combinations. It can be seen from Table VI and Table VII, that as the length of the input sentence increases, all metrics used to evaluate the model start dropping. Another observation is that as the length of input increases, the difference in the performance of the Seq2Seq model and Transformer model decreases.

5. Conclusion

In this paper, we have developed a Parts-of-Speech-based Neural Machine Translation system for European languages such as French, German, and Spanish. Two famous architectures were used for developing the system for each language, namely, Sequence-to-Sequence architecture and Transformer architecture. The Sequence-to-Sequence architecture consisted of Luong attention (for capturing essential features), Stacked Bidirectional Long Short-Term Memory Encoder, and Stacked Unidirectional Long Short-Term Memory Decoder. The Transformer model consisted of 6 Multi-head Self-attention (for capturing essential features), six layers of Dense Encoder layers, and six layers of Dense Decoder layers. The models were trained with Paracrawl, Europarl, and Tatoeba Project datasets. The models were evaluated using Precision Score, Brevity Score, Bilingual Evaluation Understudy (BLEU) Score, and Metric for Evaluation of Translation with Explicit Ordering (METEOR) Score. It was observed that for all the language combinations used in the paper, the Transformer model performed significantly better than the Luong Attention-based Sequence-to-Sequence model. The Transformer model performed significantly better because it had a higher Precision Score and Brevity Score than good Precision Score but a lower Brevity Score for Luong Attention-based Sequence-to-Sequence model.

References

[1] Ethnologue. How many languages are there in the world?, Feb 2020.
[2] Worldometer. World population (live), Oct 2020.
[3] Wikipedia contributors. List of languages by total number of speakers — Wikipedia, the free encyclopedia. [Online; accessed 21-September-2020].

[4] Wikipedia contributors. List of languages by the number of countries in which they are recognized as an official language — Wikipedia, the free encyclopedia, 2020.
[5] Wikipedia contributors. Official language — Wikipedia, the free encyclopedia, 2020.
[6] Dana Smith. At what age does our ability to learn a new language like a native speaker disappear?, May 2018.
[7] Steffy Zameo. Neural machine translation: Tips & advantages for digital translations — textmaster, May 2019.
[8] Delip Rao. The real problems with neural machine translation, Jul 2018.
[9] Wikipedia contributors. Statistical machine translation — Wikipedia, the free encyclopedia. [Online; accessed 21-September-2020].
[10] Wikipedia contributors. Neural machine translation — Wikipedia, the free encyclopedia. [Online; accessed 21-September-2020].
[11] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
[12] Wikipedia contributors. Seq2seq — Wikipedia, the free encyclopedia, 2020. [Online; accessed 21-September-2020].
[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
[14] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
[15] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.
[16] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71. Association for Computational Linguistics, November 2018.
[17] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
[18] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*, 2016.
[19] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, AN Gomez, L Kaiser, and I Polosukhin. Attention is all you need. arxiv 2017. *arXiv preprint arXiv:1706.03762*, 2017.
[20] Yongjing Yin, Jinsong Su, Huating Wen, Jiali Zeng, Yang Liu, and Yidong Chen. Pos tag-enhanced coarse-to-fine attention for neural machine translation. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 18(4), April 2019.
[21] Jan Niehues and Eunah Cho. Exploiting linguistic resources for neural machine translation using multi-task learning, 2017.
[22] Kelly, Charles. English-spanish sentences from the tatoeba project, 2020. [Online; Accessed 20 September 2020.].
[23] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. Citeseer, 2005.
[24] Paracrawl, 2018.
[25] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA, 2002. Association for Computational Linguistics.
[26] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.
[27] J. F. Kolen and S. C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. 2001.
[28] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *ArXiv*, abs/1211.5063, 2012.
[29] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. Bleu: a method for automatic evaluation of machine translation. pages 311–318, 2002.
[30] Satanjeev Banerjee and Alon Lavie. 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*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.
[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016.
[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

TABLE IV
PERFORMANCE OF THE MODELS ON THE VALIDATION SET

Translation Task	Model	BLEU Score	Precision Score	Brevity Score	METEOR Score
English-French	Seq2Seq	32.148	35.428	90.74	57.155
	Transformer	**40.929**	**41.591**	**98.408**	**66.047**
French-English	Seq2Seq	37.478	41.065	91.265	59.996
	Transformer	**40.713**	**41.898**	**97.712**	**69.469**
English-German	Seq2Seq	29.824	32.203	92.612	54.263
	Transformer	**36.925**	**37.45**	**98.6**	**62.66**
German-English	Seq2Seq	39.585	42.463	93.222	67.906
	Transformer	**42.951**	**44.34**	**96.867**	**71.483**
English-Spanish	Seq2Seq	37.478	41.065	91.265	59.996
	Transformer	**44.673**	**45.637**	**97.886**	**68.43**
Spanish-English	Seq2Seq	37.478	41.065	91.265	59.996
	Transformer	**45.551**	**46.503**	**97.953**	**72.913**

TABLE V
PERFORMANCE OF THE MODELS ON THE TESTING SET

Translation Task	Model	BLEU Score	Precision Score	Brevity Score	METEOR Score
English-French	Seq2Seq	31.654	34.67	91.301	57.056
	Transformer	**39.84**	**40.318**	**98.816**	**65.969**
French-English	Seq2Seq	37.568	**41.078**	91.456	59.968
	Transformer	**39.916**	41.069	**97.194**	**69.561**
English-German	Seq2Seq	28.742	31.012	92.681	53.466
	Transformer	**36.022**	**36.505**	**98.679**	**62.175**
German-English	Seq2Seq	38.72	41.218	93.94	67.37
	Transformer	**42.172**	**43.365**	**97.249**	**70.895**
English-Spanish	Seq2Seq	37.568	41.078	91.456	59.968
	Transformer	**44.786**	**45.648**	**98.111**	**68.176**
Spanish-English	Seq2Seq	37.568	41.078	91.456	59.968
	Transformer	**45.601**	**46.274**	**98.545**	**72.83**

TABLE VI
PERFORMANCE OF THE MODELS ON THE TESTING SET BASED ON LENGTH OF INPUT SENTENCE USING PRECISION SCORE AND BREVITY SCORE

Translation Task	Model	Precision Score					Brevity Score				
		< 10	10 - 19	20 - 29	30 - 39	40 <	< 10	10 - 19	20 - 29	30 - 39	40 <
English-French	Seq2Seq	38.175	37.533	34.231	30.21	21.925	93.551	93.377	92.045	85.777	87.877
	Transformer	**48.41**	**43.363**	**38.256**	**37.527**	**31.152**	**98.232**	**99.575**	**99.617**	**96.257**	**99.624**
French-English	Seq2Seq	42.902	41.319	**40.494**	**41.484**	35.269	86.293	92.034	92.838	90.378	85.498
	Transformer	**51.808**	**45.625**	40.194	37.601	**38.708**	**95.374**	97.588	97.305	97.187	95.571
English-German	Seq2Seq	41.618	33.662	30.292	26.98	25.319	88.071	93.94	94.27	90.383	81.093
	Transformer	**47.325**	**39.326**	**35.373**	**32.669**	**31.311**	**99.251**	**99.273**	**99.538**	**96.802**	**89.649**
German-English	Seq2Seq	53.005	45.057	39.594	34.654	24.703	91.397	94.591	95.378	91.231	83.51
	Transformer	**53.719**	**45.583**	**42.117**	**38.983**	**33.317**	**96.687**	**97.628**	**97.507**	**96.534**	**89.965**
English-Spanish	Seq2Seq	42.902	41.319	40.494	41.484	35.269	86.293	92.034	92.838	90.378	85.498
	Transformer	**51.618**	**45.694**	**44.889**	**45.45**	**43.438**	**98.498**	**99.062**	**98.39**	**96.734**	**91.922**
Spanish-English	Seq2Seq	42.902	41.319	40.494	41.484	35.269	86.293	92.034	92.838	90.378	85.498
	Transformer	**51.851**	**47.281**	**44.438**	**47.191**	**36.561**	**97.832**	**98.641**	**98.942**	**98.499**	**93.614**

TABLE VII
PERFORMANCE OF THE MODELS ON THE TESTING SET BASED ON LENGTH OF INPUT SENTENCE USING BLEU SCORE AND METEOR SCORE

Translation Task	Model	BLEU Score					METEOR Score				
		< 10	10 - 19	20 - 29	30 - 39	40 <	< 10	10 - 19	20 - 29	30 - 39	40 <
English-French	Seq2Seq	35.713	35.047	31.508	25.913	19.267	58.242	59.531	56.287	49.576	40.289
	Transformer	**47.555**	**43.178**	**38.109**	**36.123**	**31.035**	**70.647**	**67.768**	**63.325**	**60.07**	**52.671**
French-English	Seq2Seq	37.022	38.027	37.594	**37.493**	30.154	58.572	60.763	60.534	58.833	52.854
	Transformer	**49.411**	**44.524**	**39.111**	36.544	**36.994**	**74.864**	**71.656**	**67.625**	**65.431**	**65.291**
English-German	Seq2Seq	36.654	31.622	28.556	24.385	20.532	57.134	55.164	51.987	47.603	43.072
	Transformer	**46.971**	**39.04**	**35.209**	**31.624**	**28.07**	**69.052**	**63.28**	**59.566**	**56.255**	**51.57**
German-English	Seq2Seq	48.445	42.62	37.764	31.615	20.63	72.479	69.167	64.916	58.428	46.663
	Transformer	**51.939**	**44.502**	**41.068**	**37.632**	**29.973**	**76.09**	**71.663**	**68.555**	**65.062**	**55.38**
English-Spanish	Seq2Seq	37.022	38.027	37.594	37.493	30.154	58.572	60.763	60.534	58.833	52.854
	Transformer	**50.843**	**45.266**	**44.167**	**43.965**	**39.929**	**71.616**	**68.473**	**67.131**	**65.459**	**61.597**
Spanish-English	Seq2Seq	37.022	38.027	37.594	37.493	30.154	58.572	60.763	60.534	58.833	52.854
	Transformer	**50.727**	**46.638**	**43.968**	**46.483**	**34.226**	**74.82**	**73.541**	**71.668**	**71.924**	**61.994**