NEURAL MACHINE TRANSLATOR

Anonymous ACL submission

Abstract

This research paper explores the development of a neural machine translator using TensorFlow and various associated libraries. Leveraging WordCloud, Tokenizer, and LSTM-based architecture, our model demonstrates effective translation capabilities. We utilize a combination of preprocessing techniques and advanced neural network layers to enhance translation accuracy. The study incorporates insightful visualizations and evaluation metrics, including BLEU score, shedding light on the model's proficiency in language translation tasks.

1 Introduction

In an era marked by global connectivity and the relentless exchange of information, Neural Machine Translation (NMT) systems stand as pivotal tools in fostering seamless cross-language communication. As societies become increasingly interconnected, the importance of accurate and context-aware translation systems cannot be overstated.

The essence of NMT lies in its ability to comprehend and generate linguistically nuanced translations, facilitating effective communication across diverse linguistic landscapes. Modern NMT systems are expected to meet stringent requirements to be deemed effective. These requirements include a deep understanding of the semantics of the source language, the capacity to handle variations in sentence length, and the aptitude to capture subtle linguistic nuances for contextually accurate translations.

Our proposed model encompasses a three-layered encoder-decoder architecture, where the encoder effectively captures intricate patterns within the source language (English), and the decoder generates coherent sequences in the target language (German). The integration of Long Short-Term Memory (LSTM) units facilitates the modeling of sequential dependencies, and a pivotal aspect of our innovation lies in the incorporation of an Attention Layer.

The Attention Layer serves as the crux of our model, endowing it with the ability to dynamically focus on different parts of the source sequence during the decoding process. This mechanism not only enhances the model's understanding of the input sequence but also enables it to selectively weigh the importance of different words, mimicking the human cognitive process in language comprehension.

Significantly, our research contributes to the broader landscape of NMT by presenting a detailed exploration of the Attention Layer's inner workings and its impact on translation accuracy. We have meticulously trained and fine-tuned the model on a substantial English-German parallel corpus, incorporating rigorous validation and testing procedures.

Our exploration into English-German translation introduces a sophisticated NMT architecture featuring a customized Attention Layer. The attention mechanism's way of handling of extended sentences and nuanced linguistic constructs positions our model as a promising tool for generating contextually rich translations. Through this research, we aim to make a significant stride towards advancing language technology and fostering seamless cross-language communication in diverse domains.

2 Literature Review

Neural Machine Translation (NMT) has witnessed significant advancements in recent years, with state-of-the-art models demonstrating impressive translation capabilities. Earlier works have paved the way for these breakthroughs, contributing valuable insights and techniques. Some of the notable NMT models and ideas that have played pivotal roles in shaping the field are listed below.

2.1 Sequence-to-sequence (Seq2Seq) model

Proposed by Sutskever et al. (2014), this model introduced the fundamental encoder-decoder architecture, where an encoder processes the source language sequence, and a decoder generates the target language sequence. The Seq2Seq model laid the groundwork for subsequent, more sophisticated architectures.

2.2 Attention Mechanism

It was a key innovation in NMT introduced by Bahdanau et al. (2014). While not the first attention-based model, their work significantly improved the alignment and translation quality. The attention mechanism allows the model to focus on different parts of the source sequence during decoding, addressing the limitation of Seq2Seq models in handling long input sequences.

2.3 Global Attention Mechanism:

In addressing the challenge of rare words, Luong et al. (2015) proposed the Global Attention Mechanism. This mechanism enhances the alignment process by considering the entire source sequence at each decoding step, allowing the model to capture global context and improve translation performance on rare or out-of-vocabulary words.

2.4 Convolutional Sequence-to-Sequence model

Another noteworthy work is the introduction of convolutional neural networks (CNNs) to NMT by Gehring et al. (2017). Their Convolutional Sequence-to-Sequence model demonstrated competitive performance while reducing the computational demands compared to recurrent neural network (RNN)-based architectures. This highlighted the potential of alternative neural network architectures for sequence-to-sequence tasks.

3 Dataset and Preprocessing:

3.1 Data Description

The dataset encompasses a diverse collection of parallel text data, aligning English sentences with their corresponding German translations. It comprises two main columns: "English" and "German." Each row in the dataset represents a pair of parallel sentences, where the "English" column contains sentences in English, and the "German" column holds their corresponding translations in German.

3.2 The data preprocessing

steps are essential for preparing the text data for training a Neural Machine Translation model. The steps carried out were:

3.2.1 Reading Data:

The Reading Data step involved loading parallel training data from two text files—one for English input sentences and the other for German output sentences. This was the foundational step to access the raw text data, allowing further processing.

Statistic	eng_len	ger_len
Count	9999.000000	9999.000000
Mean	140.540754	145.752675
Standard Deviation	79.559802	81.221845
Minimum	23.000000	24.000000
25% Percentile	83.000000	87.000000
50% Percentile	121.000000	127.000000
75% Percentile	179.000000	184.000000
Maximum	649.000000	701.000000

Figure 1: Statistics for English and German sequence lengths

3.2.2 Text Cleaning :

In the Text Cleaning step, we have employed a function to standardize the text by converting it to lowercase and removing non-alphanumeric characters using a regular expression. This process was crucial for achieving consistency in the data. By ensuring that all text was in lowercase, the model became case-insensitive, and removing non-alphanumeric characters helped simplify the vocabulary, making it easier for the model to generalize.

3.2.3 Exploratory Data Analysis (EDA):

Exploratory Data Analysis was a crucial step in understanding the characteristics of the training data. The EDA function in the code performed several key tasks to ensure data quality. Firstly, it created a DataFrame (df) using a subset of the training data, allowing for a manageable exploration and training(as handling very large datasets would take a long time on current systems). By checking for missing values (na) and duplicated values, the function ensured that the data was complete and free from redundant samples.

The distribution of sentence lengths for both the input and output languages was visualized using histograms. This step was significant as it provided insights into the variability of sentence lengths within the dataset. Understanding the range and distribution of sentence lengths was essential for designing a model architecture that could effectively handle sentences of different lengths. It aided in determining the appropriate sequence length for padding later in the preprocessing pipeline.

The statistical summary printed at the end of the EDA step, showing the mean, standard deviation, minimum, and maximum lengths of sentences, offered a quantitative understanding of the dataset's structure. This information was valuable for config-

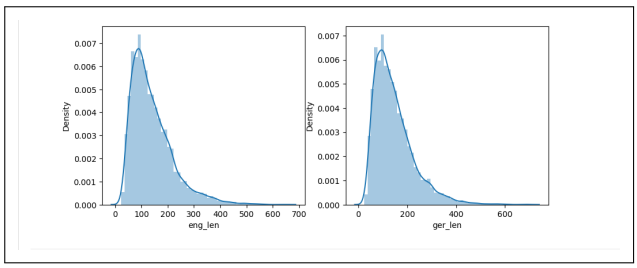

Figure 2: Histogram for Sentence Lengths

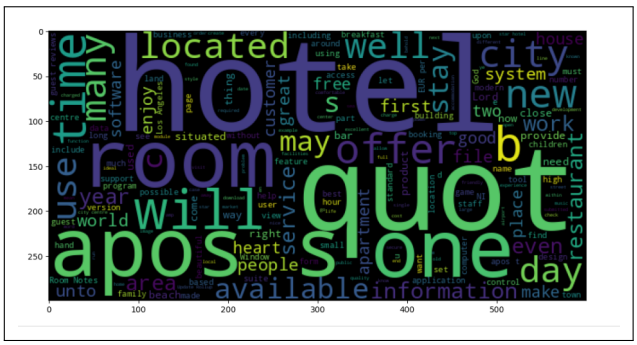

Figure 3: Word Cloud for English

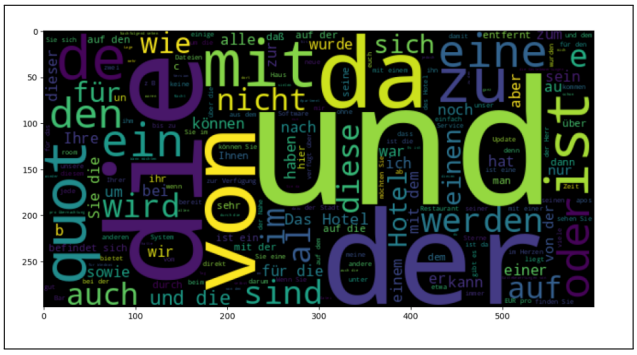

Figure 4: Word Cloud for German

uring the model architecture and hyperparameters, ensuring the neural network could handle sentences of varying lengths effectively.

3.2.4 Word Cloud Visualization:

The Word Cloud Visualization step offered a visual representation of the most frequent words in a given DataFrame column. The visualization is generated using a graphical arrangement of words in varying sizes, where the size of each word corresponds to its frequency or importance. Thus, the more frequently a word appeared in the dataset, the larger and more prominent it appeared in the Word Cloud.

Understanding the most frequent words was crucial for several reasons. Firstly, it aided in identifying the predominant terms within the dataset, offering an at-a-glance summary of the common vocabulary. This visual representation was particularly helpful in discerning patterns or recurring themes that could be crucial for the subsequent training of a Neural Machine Translation model.

Moreover, the word cloud served as an intuitive tool for data exploration, allowing us to quickly identify noteworthy terms or potential outliers. It acted as a visual aid that could uncover unique characteristics of the data, providing insights that were not immediately apparent through quantitative analyses alone.

In the context of NMT, where capturing the nuances of language is paramount, the word cloud visualization shed light on the distribution and prevalence of specific terms. This understanding was beneficial for making informed decisions during preprocessing, such as determining whether certain words required special handling or consideration, or if there are unexpected artifacts that merit further investigation.

3.2.5 Text Tokenization and Padding:

Text tokenization was a critical step in preparing text data for neural network training. The preprocess data function used the Keras Tokenizer class to convert the cleaned input and output sentences into numerical sequences. Tokenization assigned a unique numerical index to each word in the vocabulary, allowing the neural network to work with numerical data.

The addition of '<START>' and '<END>' tokens to the output sentences served a specific purpose in the model as it was 'sequence to sequence' in nature. These tokens indicated the beginning and end of each translation, providing the model with explicit cues about sentence boundaries. This was crucial for the model to learn the structure of sentences and generate accurate translations.

Padding, achieved through the pad sequences function, ensured that all token sequences had the same length. In deep learning models, input sequences typically need to be of uniform length for efficient batch processing. Padding ensured that shorter sentences were filled with padding tokens, while longer sentences were truncated to the specified maximum length. This step was vital for creating consistent input data, enabling the model to process batches of sentences in parallel during training. It facilitated the creation of a fixed-size input tensor, streamlining the training process and improving computational efficiency.

4 Methodology

In this project , comprehensive neural machine translator (NMT) model is developed using TensorFlow and Keras. The architecture follows the sequence-to-

Library/Module	Purpose
wordcloud	Word cloud visualization
tensorflow.keras.preprocessing.sequence	Sequence padding
tensorflow.keras.preprocessing.text	Text Tokenization
tensorflow.keras.layers	Neural network layers
tensorflow.keras.callbacks	Callbacks for model training
tensorflow.keras.models	Model building and loading
numpy	Numerical operations
pandas	Data manipulation and analysis
seaborn	Data visualization
matplotlib.pyplot	Plotting
warnings, re	Handling warnings and regular expressions
tensorflow	Deep learning framework
tqdm.notebook	Progress bar in Jupyter notebooks
tensorflow.keras.backend	Backend operations
nltk.translate.bleu_score	BLEU score computation

Table 1: List of Libraries and Modules

sequence paradigm with attention mechanism integration, aiming to effectively translate text from one language to another. The encoder-decoder framework is a key component of this model. The encoder processes input sequences, represented as word embeddings, through multiple LSTM layers with varying dropout rates. These dropout layers contribute to the model's generalization ability by preventing overfitting during training.

The attention mechanism, implemented as a custom layer, enhances the model's translation capabilities by allowing it to focus on different parts of the input sequence during decoding. The attention layer dynamically computes attention scores, emphasizing relevant information. The decoder, equipped with its own LSTM layers and dropout, generates the output sequence while considering attention-weighted context vectors from the encoder. A TimeDistributed Dense layer with a softmax activation produces the final translation probabilities for each word in the output sequence.

Training the model involves compiling it with a suitable loss function (sparse categorical crossentropy) and optimizer (RMSprop). The training dataset is split into training and validation sets, and early stopping, learning rate reduction, and model checkpointing are employed as callbacks during training to enhance performance and prevent overfitting. The model is trained on the provided English and German text sequences, optimizing for accuracy.

This NMT model encapsulates a sophisticated architecture that learns intricate patterns and dependencies between source and target languages. It leverages the power of LSTM layers, attention mechanisms, and dropout regularization to achieve accurate and contextually relevant translations. The training process is monitored using various callbacks, ensuring the model's generalization and preventing it from learning noise in the data. Overall, this code represents a thoughtful and effective approach to building a neural machine translator for language translation tasks.

5 MODEL ARCHITECTURE OVERVIEW

5.1 Encoder :

The encoder in the neural machine translator model consists of a series of Long Short-Term Memory (LSTM) layers. These layers are responsible for processing the input sequence in a sequential manner, capturing the temporal dependencies and contextual information of the source language. The embedding layer at the beginning converts input words into dense vectors, allowing the model to learn and represent the semantic meaning of each word. Each subsequent LSTM layer further refines this representation by considering the context from the preceding layer. The dropout is employed within each LSTM layer to prevent overfitting during training. This technique randomly deactivates a fraction of the neurons, forcing the model to learn more robust and generalized features.

5.2 Decoder :

The decoder also employs LSTM layers to process the target sequence. The decoder's initial state is

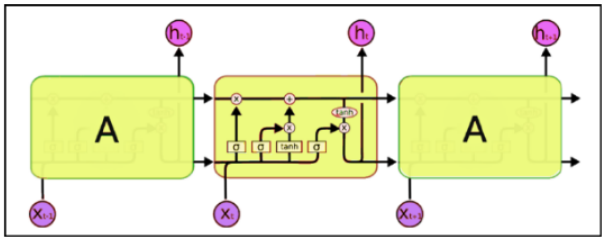

Figure 5: Simple LSTM Architecture

set using the final states of the encoder, enabling the model to leverage the learned representation of the source sequence. Similar to the encoder, the dropout is applied to the decoder's LSTM layers to enhance generalization. The decoder LSTM generates an output sequence while considering the attention-weighted context vectors obtained from the attention mechanism. This allows the model to focus on different parts of the source sequence during the generation of each target word. The final output sequence is produced by a TimeDistributed Dense layer, applying a softmax activation function to ensure the output represents a valid probability distribution over the German vocabulary.

5.3 Dropout and Activation Functions

Dropout is employed in both the encoder and decoder layers to mitigate overfitting during training. By randomly deactivating a fraction of neurons, dropout prevents the model from relying too heavily on specific features, promoting a more robust and generalized representation of the input data.

The activation functions used in the LSTM layers include hyperbolic tangent (tanh) for the main activation and sigmoid for the recurrent activation. Tanh is chosen for its ability to model complex relationships and alleviate the vanishing gradient problem, while sigmoid is suitable for controlling the flow of information within the recurrent units. These choices collectively contribute to the model's ability to capture intricate patterns in the source and target languages, facilitating effective translation.

5.4 Attention Layer

The Attention Layer plays a pivotal role in improving the performance of neural machine translation models by allowing the model to selectively focus on different parts of the input sequence during the decoding process. The attention mechanism is crucial for capturing long-range dependencies and aligning source and target sequences effectively.

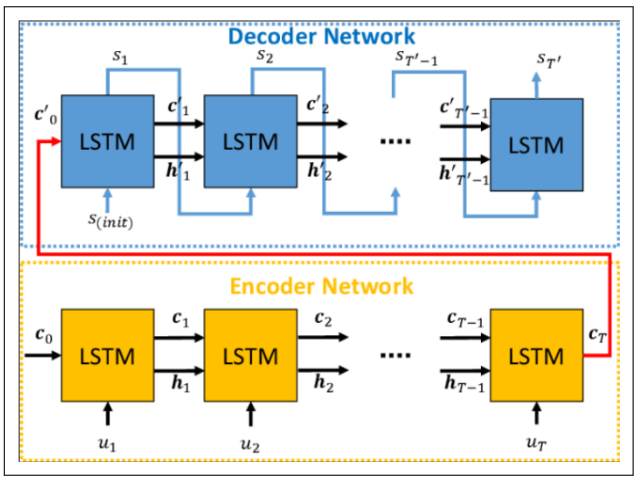

Figure 6: Encoder-Decoder LSTM Architecture

5.4.1 Initialization:

The attention layer begins by initializing trainable weight variables. These weights are crucial for computing attention scores and represent the learned relationships between different parts of the input and output sequences.

5.4.2 Energy Computation:

The attention mechanism computes an energy score for each element in the input sequence, considering the current state of the decoder. The energy score is calculated through a series of operations involving the encoder's output sequence , decoder's output sequence, and the trainable weights . The computation involves a combination of linear transformations, hyperbolic tangent activations, and softmax functions.

5.4.3 Context Vector Computation:

Following the energy computation, the attention layer calculates a context vector for each element in the input sequence. This context vector is a weighted sum of the encoder's output sequence, where the weights are determined by the attention scores obtained in the energy computation step. The context vector captures the relevant information from the input sequence, emphasizing elements that are deemed more important by the attention mechanism.

5.4.4 Stateless Attention Computation:

Notably, the attention mechanism is designed to be stateless. It does not maintain states between steps during energy and context computations, simplifying the implementation. The layer utilizes a fake state for the RNN step function, ensuring that attention is recalculated at each decoding step, allowing the model to dynamically adjust its focus based on the evolving context.

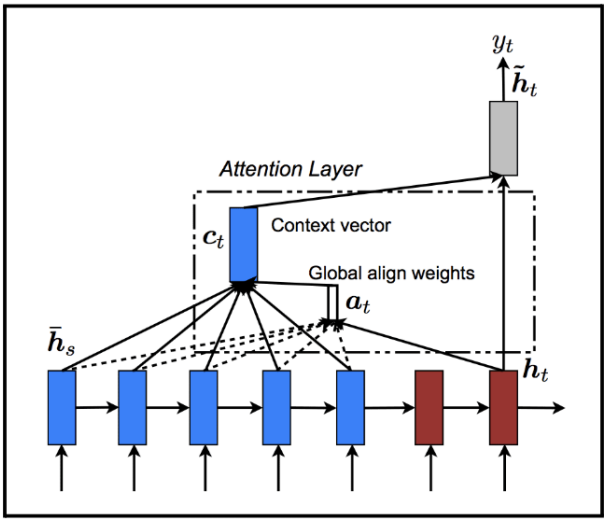

Figure 7: Attention Layer Mechanism

Hyperparameter	Description & Explanation	Optimal Value
Maximum English Sequence Length	The maximum length of input sequences in English. Longer sequences may be truncated or padded.	649
Maximum German Sequence Length	The maximum length of output sequences in German. Longer sequences may be truncated or padded.	649
Latent Dimension	The dimensionality of the hidden state in LSTM layers. A higher value allows the model to capture more complex patterns but increases computational cost.	30
Embedding Dimension	The dimensionality of word embeddings. Higher values may capture more semantic information but also increase the model's complexity.	10

Figure 8: Neural Machine Translation Hyperparameter

5.4.5 Final Outputs:

The final outputs of the attention layer consist of context vectors and attention scores. These outputs are integral in enhancing the decoder's understanding of the input sequence, providing enriched information for generating accurate and context-aware translations.

In summary, the attention layer facilitates a dynamic and context-aware translation process, allowing the model to selectively attend to relevant parts of the source sequence at each decoding step. This mechanism significantly improves the model's ability to capture nuanced relationships and dependencies between words in different positions of the input and output sequences.

5.5 Concatenation Layer:

The outputs from the decoder LSTM and the attention layer are concatenated to create a fused representation. This merged information captures both the decoder's intrinsic output and the relevant context from the source sequence. The concatenation ensures that the model considers the attention-weighted context while generating each element of the output sequence.

5.6 TimeDistributed Dense Layer:

The final layer is a TimeDistributed Dense layer, which produces a probability distribution over the German vocabulary for each time step. The softmax activation function ensures that the model outputs a valid probability distribution. This layer is crucial for the generation of the final translated sequence.

5.7 Model Summary:

The complete model is an end-to-end sequence-to-sequence neural machine translation architecture. It seamlessly integrates the encoder, attention layer, and decoder, providing a comprehensive framework for language translation tasks. The model's summary highlights the connectivity between different components and the flow of information, showcasing its ability to learn and generate translations effectively.

6 RESULTS

In our neural machine translation study, we present comprehensive results for both English-to-German and German-to-English translation tasks. The performance of our models is assessed using key metrics, including loss and accuracy graphs, which illustrate the training process and convergence of the models. Additionally, we provide insights into the translation quality by showcasing input texts alongside the predicted and actual output texts. The effectiveness of our translations is quantitatively evaluated using BLEU scores, a widely-accepted metric in machine translation that measures the similarity between predicted and reference texts. Our results not only demonstrate the training dynamics but also offer a nuanced understanding of the model's proficiency in capturing linguistic nuances and generating contextually accurate translations for both language directions.

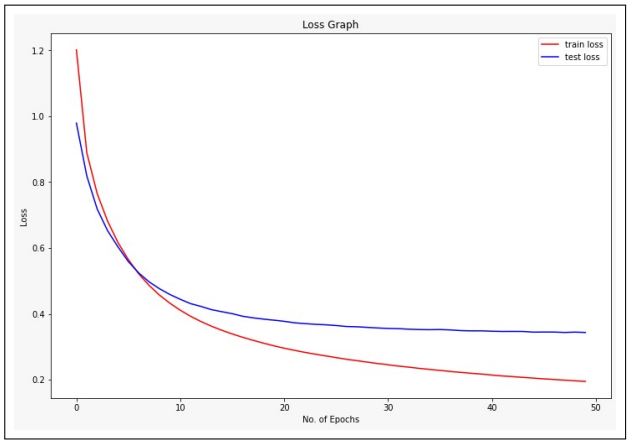

Figure 9: Loss Graph

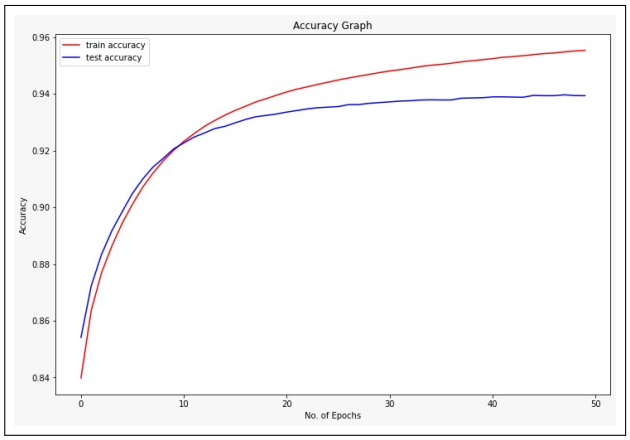

Figure 10: Accuracy Graph

Figure 11: English to German translation with BLEU Score

Figure 12: German to English translation with BLEU Score

7 Conclusion

In conclusion, the presented neural machine translation model exhibits a comprehensive architecture designed to facilitate the translation of English sentences into German. The model encompasses a multi-layered encoder-decoder structure, incorporating attention mechanisms to capture contextual information effectively. Despite the observed challenges in achieving optimal translation results in the preliminary stages due to lesser data, the model's significance for future research and development is noteworthy.

The use of recurrent neural networks, specifically Long Short-Term Memory (LSTM) cells, in both the encoder and decoder components allows the model to capture long-range dependencies in the input sequences. The attention mechanism further enhances the model's ability to focus on relevant parts of the source sentence during the translation process, addressing issues related to information loss over longer sequences.

The dropout layers strategically placed within the architecture contribute to regularization, mitigating the risk of overfitting and enhancing the model's generalization capabilities. Additionally, the embedding layers provide a meaningful representation of words in a continuous vector space, allowing the model to learn intricate linguistic nuances.

Despite the initial modest translation performance, the potential significance of this model lies in its adaptability and extensibility. Future endeavors could explore fine-tuning hyperparameters, experimenting with different attention mechanisms, or incorporating more advanced architectures such as Transformer models. Furthermore, the model's training process could benefit from larger and more diverse datasets to improve its language understanding and translation proficiency.

The continual advancement in neural machine

translation holds promise for overcoming current limitations. The presented model, with its foundational architecture and room for enhancement, can serve as a valuable baseline for researchers and practitioners alike. Future investigations can build upon this foundation, exploring novel techniques and architectures to propel the development of more accurate and context-aware translation models. As the field evolves, the lessons learned from this model can contribute to the ongoing discourse on improving the efficiency of neural machine translation systems.

References

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Gehring, J., Auli, M., Grangier, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Qiao, Y., Hashimoto, K., Eriguchi, A., Wang, H., Wang, D., Tsuruoka, Y., & Taura, K. (2020). Parallelizing and optimizing neural Encoder–Decoder models without padding on multi-core architecture. *Future Generation Computer Systems*, 108, 1206–1213. doi:10.1016/j.future.2018.04.070

Xia, Y., He, T., Tan, X., Tian, F., He, D., & Qin, T. (2019, July). Tied transformers: Neural machine translation with shared encoder and decoder. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 5466–5473. doi:10.1609/aaai.v33i01.33015466

Tufano, M., Pantiuchina, J., Watson, C., Bavota, G., & Poshyvanyk, D. (2019, May). On learning meaningful code changes via neural machine translation. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 25-36). IEEE. 10.1109/ICSE.2019.00021

Nishimura, Y., Sudoh, K., Neubig, G., & Nakamura, S. (2019). Multi-source neural machine translation with missing data. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 28*, 569–580. doi:10.1109/TASLP.2019.2959224

Soltau, H., Liao, H., & Sak, H. (2016). Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975*.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*. doi:10.18653/v1/D15-1166

Xia, Y. (2020). Research on statistical machine translation model based on deep neural network. *Computing, 102(3)*, 643–661. doi:10.1007/s00607-019-00752-1

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*. doi:10.3115/v1/W14-4012

Chaudhary, J. R., & Patel, A. C. (2018). Bilingual machine translation using RNN based deep learning. *International Journal of Scientific Research in Science, Engineering and Technology*, 4(4), 1480–1484.

Kumar, K. C., Aswale, S., Shetgaonkar, P., Pawar, V., Kale, D., & Kamat, S. (2020, February). A Survey of Machine Translation Approaches for Konkani to English. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)* (pp. 1-6). IEEE.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chen, K., Zhao, T., Yang, M., Liu, L., Tamura, A., Wang, R., Utiyama, M., & Sumita, E. (2017). A neural approach to source dependence based context model for statistical machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(2), 266–280. doi:10.1109/TASLP.2017.2772846

Zhang, B., Xiong, D., Su, J., & Duan, H. (2017). A context-aware recurrent encoder for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(12), 2424–2432. doi:10.1109/TASLP.2017.2751420

Singh, S. P., Kumar, A., Darbari, H., Singh, L., Rastogi, A., & Jain, S. (2017, July). Machine translation using deep learning: An overview. In *2017 International Conference on Computer, Communications and Electronics (Comptelix)* (pp. 162-167). IEEE. doi:10.1109/COMPTELIX.2017.8003957

Chaudhary, J. R., & Patel, A. C. (2018). Bilingual machine translation using RNN based deep learning. *International Journal of Scientific Research in Science, Engineering and Technology*, 4(4), 1480–1484.

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_1 (InputLayer)        [(None, 649)]             0         []

 embedding (Embedding)       (None, 649, 10)           194290    ['input_1[0][0]']

 lstm (LSTM)                 [(None, 649, 30),         4920      ['embedding[0][0]']
                              (None, 30),
                              (None, 30)]

 lstm_1 (LSTM)               [(None, 649, 30),         7320      ['lstm[0][0]']
                              (None, 30),
                              (None, 30)]

 input_2 (InputLayer)        [(None, None)]            0         []

 lstm_2 (LSTM)               [(None, 649, 30),         7320      ['lstm_1[0][0]']
                              (None, 30),
                              (None, 30)]

 embedding_1 (Embedding)     (None, None, 10)          298440    ['input_2[0][0]']

 lstm_3 (LSTM)               [(None, 649, 30),         7320      ['lstm_2[0][0]']
                              (None, 30),
                              (None, 30)]

 lstm_4 (LSTM)               [(None, None, 30),        4920      ['embedding_1[0][0]',
                              (None, 30),                         'lstm_3[0][1]',
                              (None, 30)]                         'lstm_3[0][2]']

 attention_layer (Attention  ((None, None, 30),        1830      ['lstm_3[0][0]',
 Layer)                       (None, None, 649))                  'lstm_4[0][0]']

 concat_layer (Concatenate)  (None, None, 60)          0         ['lstm_4[0][0]',
                                                                  'attention_layer[0][0]']

 time_distributed (TimeDist  (None, None, 29844)       1820484   ['concat_layer[0][0]']
 ributed)

=================================================================================================
Total params: 2346844 (8.95 MB)
Trainable params: 2346844 (8.95 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________________________
None
```

Figure 13: Model Summary