Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

A Comprehensive Analysis of Modern NLP Architectures: Bridging RNNs and Transformers for Enhanced Language Understanding

Gauraangi Praakash[1] , Pooja Khanna[2]

[1]Department of Computer Science and Engineering Amity School of Engineering and Technology Lucknow, Amity University Uttar Pradesh Lucknow, India gauraangi25@gmail.com
[2]Department of Computer Science and Engineering Amity School of Engineering and Technology Lucknow, Amity University Uttar Pradesh Lucknow, India pkhanna@lko.amity.edu

This study explores the application of Recurrent Neural Networks (RNNs) for machine translation. Two distinct models are analyzed: a simple RNN and an RNN enhanced with embeddings. To preprocess the data, word tokenization and Byte Pair Encoding (BPE) techniques are used, which help manage the input text efficiently. Model 1 utilizes a basic RNN architecture combined with word tokenization and BPE, while Model 2 incorporates embedding layers to improve the semantic representation of words. Through a comparative analysis, it is observed that Model 2, with its use of embeddings, demonstrates superior performance in terms of both translation accuracy and overall efficiency. BPE plays a significant role in both models, effectively addressing challenges such as out-of-vocabulary words and enhancing the translation of longer and more complex sentences. The results show that embedding layers significantly improve the RNN's ability to capture word meanings and produce more accurate translations, particularly for morphologically complex languages like French. Furthermore, the study discusses potential future improvements, such as the integration of attention mechanisms, which could further enhance the model's ability to focus on relevant parts of the input sequence and boost performance in more challenging translation tasks. Overall, this research highlights the effectiveness of RNNs, embeddings, and BPE in machine translation and provides a foundation for future exploration into more advanced neural translation models.

Keywords— Recurrent Neural Networks (RNN), Word Tokenization, Byte Pair Encoding (BPE), Language Translation, Embeddings, Neural Machine Translation (NMT), Sequence-to-Sequence Models.

I. INTRODUCTION

Language translation is a pivotal task in natural language processing (NLP) that has grown in importance with the rise of global communication and information sharing. The field of Natural Language Processing (NLP) has undergone dramatic transformation with the advent of attention mechanisms and transformer architectures. The fundamental breakthrough came with

the realization that 'attention is all you need' [1], shifting the paradigm from traditional sequential processing. This advancement built upon earlier work in encoder-decoder architectures [2], which demonstrated the potential of neural approaches. The introduction of joint learning mechanisms for alignment and translation [3] further enhanced model capabilities, leading to significant improvements in translation quality. Historically, language translation relied on rule-based systems, which were highly rigid, or statistical machine translation (SMT) methods, which employed probabilistic models of word alignment. While these systems played a crucial role in early translation tasks, they were limited by their inability to fully capture the intricacies of human language. This changed with the advent of neural machine translation (NMT), where deep learning models, particularly Long Short-Term Memory (LSTM) and Sequence-to-Sequence (Seq2Seq) architectures, have demonstrated substantial improvements in translation quality and fluency.

The Sequence-to-Sequence architecture, proposed by Sutskever et al. (2014), laid the foundation for contemporary NMT systems. By mapping sequences from a source language to a target language, Seq2Seq models improved upon traditional methods by handling variable-length input and output sequences. The introduction of the attention mechanism by Bahdanau et al. (2015) further advanced NMT by allowing models to focus on different parts of the input sentence during translation, which is especially important for long and complex sentences. While these developments have enhanced machine translation, several challenges remain.

This research focuses on addressing these challenges using a Long Short-Term Memory (LSTM)-based Sequence-to-Sequence model. The study explores the use of transfer learning to enhance translation for low-resource languages and advanced attention mechanisms to improve contextual understanding. The primary aim is to develop a robust model that can deliver high-quality translations across multiple languages, particularly focusing on low-resource language pairs.

II. MACHINE LEARNING IN LANGUAGE TRANSLATION

Machine learning (ML) has revolutionized numerous domains, and one of its most impactful applications is in language translation. Language, with its complexities of syntax, semantics, and contextual nuances, presents a unique challenge for computational systems. Traditional methods of translation relied heavily on rule-based systems and statistical models, but these approaches often fell short in capturing the subtleties of human language. With the advent of machine learning, particularly deep learning, language translation has undergone significant advancements. This section explores the integration of machine learning techniques into language translation, highlighting its progress, challenges, and innovations.

A. The Evolution of Language Translation

Language translation has long been a critical area in computational linguistics. Early attempts at machine translation (MT) involved rule-based systems that used predefined linguistic rules to translate text from one language to another. These systems, while effective in some cases, struggled with languages that had significant syntactic and grammatical differences. They were also limited by their inability to adapt to new language patterns without manual intervention. The foundation of modern NLP was established through sequence-to-sequence learning [4], which demonstrated unprecedented capabilities in handling variable-length input and output

sequences. This was further enhanced by bidirectional recurrent neural networks [5], enabling models to capture context from both directions. The introduction of Long Short-Term Memory (LSTM) networks [6] addressed the critical challenge of long-term dependencies, while supervised and semi-supervised approaches using LSTM for region embeddings [7] showed promising results in text categorization.

The shift toward statistical machine translation (SMT) in the 1990s marked a significant advancement. SMT systems relied on large parallel corpora—datasets that contained aligned sentences in two languages—to calculate the probability of a word or phrase in one language translating into another. By analyzing these probabilities, SMT systems could generate translations based on statistical likelihood rather than strict rules. However, these systems still faced challenges with long-range dependencies and context-sensitive translations, often producing outputs that were grammatically correct but lacked fluency.

III. LITERATURE SURVEY

The development of machine translation (MT) has seen several advancements, evolving from rule-based systems to statistical methods and, most recently, neural networks. Early approaches like Rule-Based Machine Translation (RBMT) relied on manually crafted linguistic rules and bilingual dictionaries. While RBMT systems could handle simple translations, they were often rigid and lacked scalability across different languages and language structures. The introduction of Statistical Machine Translation (SMT) marked a significant improvement, using probabilistic models to align words between languages based on large parallel corpora. However, SMT systems, such as the phrase-based SMT introduced by Koehn et al. in 2003, struggled with long-range dependencies in sentences, leading to grammatically incorrect and contextually awkward translations. Neural Machine Translation (NMT) emerged as a breakthrough with the introduction of the Sequence-to-Sequence (Seq2Seq) model by Sutskever et al. in 2014. Unlike previous methods, Seq2Seq models utilize deep learning to process variable-length input-output sequences. This model consists of an encoder that processes the input sequence and generates a context vector summarizing the entire sentence, followed by a decoder that produces the translated output. While the Seq2Seq model greatly improved the fluency of translations, it faced challenges with long and complex sentences where information often became compressed in the fixed-length context vector. A significant advancement came with the introduction of the attention mechanism by Bahdanau et al. in 2015. Instead of relying solely on a single context vector, the attention mechanism allows the model to focus on specific parts of the input sequence while generating each word in the output. This improvement made translation models much more effective at handling longer sentences, improving overall accuracy and fluency. The attention mechanism revolutionized NMT, as it enabled models to better capture dependencies between words in a sentence. Long Short-Term Memory (LSTM) networks were also key in improving machine translation tasks. Introduced by Hochreiter and Schmidhuber in 1997, LSTMs addressed the vanishing gradient problem in Recurrent Neural Networks (RNNs) by preserving information across longer sequences. LSTMs enhanced Seq2Seq models by capturing long-range dependencies more effectively, making them particularly useful for translating longer sentences and handling complex grammatical structures. LSTM-based models have been incorporated into large-scale systems, such as Google's Neural Machine Translation (GNMT) system, which replaced Google's

previous SMT system. This implementation led to substantial improvements in translation quality, highlighting the effectiveness of LSTMs in NMT. The introduction of the Transformer model by Vaswani et al. in 2017 marked another major shift in NMT. The Transformer model discarded recurrence and relied entirely on self-attention mechanisms, enabling parallel processing of the entire input sequence. This design led to faster training times and better performance on long-range dependencies. While the Transformer model has become the dominant architecture in NMT, LSTM-based Seq2Seq models still have relevance, particularly in low-resource translation tasks. Studies like those by Johnson et al. (2017) and Zoph et al. (2016) demonstrated that multilingual training and transfer learning can improve translation quality for low-resource language pairs. However, challenges still persist in translating low-resource languages, where large parallel corpora are unavailable. While high-resource languages benefit from massive datasets, low-resource language pairs suffer from insufficient training data, resulting in degraded translation quality. This issue has been addressed in studies like Zoph et al. (2016) and Aharoni et al. (2019),
which explored multilingual training and zero-shot translation to tackle the scarcity of parallel corpora. These studies showed that transfer learning could mitigate some of the issues associated with low-resource languages, but there is still a significant performance gap between high-resource and low-resource language translations. Another major gap in NMT systems is maintaining contextual accuracy. While attention
mechanisms have improved contextual understanding, current models often struggle with translating idiomatic expressions or context-sensitive phrases, leading to unnatural or incorrect translations. Chen et al. (2019) and Zhang et al. (2020) emphasized the need for better attention mechanisms or external knowledge sources to enhance the contextual understanding in NMT models. This study aims to address these challenges by exploring the use of LSTM-based Seq2Seq models, enhanced by transfer learning, to improve translation for low-resource languages. Additionally, the study investigates the use of advanced attention mechanisms to improve contextual understanding. The methodology leverages multilingual datasets and data augmentation techniques to increase the availability of parallel corpora for low-resource language pairs. By combining these approaches,
the study aims to develop a more robust and accurate translation model, particularly for languages
where data is scarce.

IV. METHODOLGY

This study focuses on developing a machine translation system using Recurrent Neural Networks (RNN) within a Sequence-to-Sequence (Seq2Seq) architecture, enhanced by an attention mechanism. The aim is to create an effective translation model that can address the challenges of translating low-resource languages by incorporating transfer learning and efficient preprocessing techniques. RNNs are used to process sequential data, capturing the temporal dependencies between words, while the attention mechanism helps the model focus on relevant parts of the input
sequence, ensuring more accurate translations.

The workflow of the project includes several stages: data collection, preprocessing, model design, training, validation, and evaluation. Special attention is given to handling the complexities of language structure across different languages. The RNN-based Seq2Seq

model, enhanced with attention, is designed to capture the relationships between words in a sentence, improving contextual accuracy. Additionally, transfer learning is employed to optimize performance on low-resource languages by utilizing knowledge from pre-trained models on high-resource languages.

A. Data Collection

For this study, the dataset comprises English and French sentence pairs, which are crucial for training the machine translation model. The data is collected from two text files, English_data.txt and French_data.txt, stored in the project repository. The process starts with reading these files and loading the sentences into Python lists: english_sentences and french_sentences. Each sentence is processed to remove any trailing spaces, convert the text to lowercase, and eliminate punctuation marks. This preprocessing ensures that the dataset is clean and uniform, providing a solid foundation for further analysis. For example, sentences like “New Jersey is sometimes quiet during autumn and it is snowy in April” and its French counterpart “New Jersey est parfois calme pendant l’automne et il est neigeux en avril” are included in the dataset.

B. Tokenization

Tokenization is a crucial step in preparing text data for machine learning models. In this study, Keras's Tokenizer is used to convert text into numerical sequences, which are easier for the modelto process. The tokenize function initializes a Tokenizer instance and fits it on the English and French sentences separately. This step creates a dictionary mapping each word to a unique integerindex. The text data is then transformed into sequences of integers where each integer correspondsto a specific word in the vocabulary. For instance, the sentence “New Jersey is sometimes quiet during autumn” would be converted into a sequence of integers based on the tokenizer’s dictionary.This numerical representation of text is essential for feeding the data into the Seq2Seq model.

Fig.1. Tokenization

Fig. 1 illustrates a step-by-step tokenization process of the sentence "We’re moving to L.A.!" for natural language processing (NLP) applications. The process starts by splitting the original sentence into smaller units, or tokens, based on whitespace. In this case, the sentence is initially broken into the following tokens: "We’re", "moving", "to", "L.A.!". After this initial segmentation, the process continues by handling specific cases, such as prefixes and suffixes, where "We’re" is further split into "We" and "’re". The next step addresses exceptions, ensuring that certain words and punctuation marks are separated correctly. For instance,

"L.A.!" is split into "L.A." and "!", while still retaining the connection between the abbreviation "L.A." and the punctuation. The use of a diagram also emphasizes an uniform handling of special characters and exceptions like breaking the suffix “re”, but keeping the punctuation "!" in a separate token. Such hierarchical breakdowns show a proper approach for tokenization, ensuring even complex structures like contractions or abbreviations are handled properly, so that text can be processed in the best form for NLP applications like machine translation or text classification. The final output of the process is a series of tokens: "We", "'re", "moving", "to", "L.A.", "!", which ensures all components of meaning in a sentence are isolated for further analysis.

C. Padding

Padding ensures that all sequences in the dataset have the same length, which is necessary for neural networks that require consistent input dimensions. The pad function is employed to add zeros to sequences, making them uniform in length. This function uses Keras's pad_sequences to pad sequences to the length of the longest sequence in the dataset. By setting the maxlen parameter to the length of the longest sequence, the function ensures that shorter sequences are extended with zeros at the end. This padding process is crucial for handling variable-length sentences and maintaining a uniform input shape for the model. For example, if the longest sentence in the dataset has 10 tokens, all sentences are padded to this length, allowing the RNN model to process them effectively.

D. Data Pre-processing of Input Data

RNN models require numerical input, so text data must be preprocessed into sequences of integers before feeding it into the model. This involves several key steps: tokenization, padding, and handling out-of-vocabulary (OOV) terms. The preprocess function is designed to handle these tasks efficiently. Initially, the text data is tokenized into sequences of integers using the tokenize method, which converts each word into a unique integer ID based on a vocabulary dictionary. Subsequently, the pad function is applied to ensure that all sequences have the same length, which is essential for RNN models that require uniform input dimensions.

The preprocess function also reshapes the label sequences to meet the requirements of Keras’s sparse_categorical_crossentropy loss function, which expects labels to be in three dimensions. This ensures compatibility with the model's loss computation during training. The training and testing datasets are split using train_test_split, providing separate datasets for model training and evaluation. After preprocessing, the maximum sequence lengths and vocabulary sizes for both English and French datasets are calculated, which are crucial for configuring the model architecture. Handling out-of-vocabulary (OOV) terms is another important aspect of data preprocessing. OOV terms refer to words that are not present in the training vocabulary. The tokenization process for the unknown sentences may result in empty values in the tokenized output, indicating these OOV terms. For instance, if a sentence like "he dislikes grapefruit limes and lemons" contains words not present in the training vocabulary, the resulting tokenized sequence may include empty values representing these unknown terms. Proper handling of OOV terms is necessary to ensure that the model can deal with unseen words during translation. The system implementation follows proven approaches from large-scale neural machine translation systems [16], incorporating specialized

mechanisms for handling rare events [17]. We utilized the OpenNMT framework [18] for development and testing, ensuring reproducibility and standardization.

The preprocessing steps ensure that the input data is formatted correctly for RNN models, enabling effective training and evaluation. By converting text to numerical sequences, padding them to a consistent length, and addressing OOV terms, the preprocessing process prepares the data for the subsequent stages of model training and translation tasks.

E. Models

This section outlines the architecture and tokenization techniques used in two distinct RNN models for English-to-French translation. This approach builds upon effective attention-based methods [12] that revolutionized neural machine translation. The breakthrough of BERT's bidirectional transformers [13] provided the foundation for contextual understanding. We utilize global vectors for word representation [14] to capture semantic relationships, implementing the transformer architecture [15] which eliminates the need for recurrent processing. Model 1 utilizes traditional word tokenization, while Model 2 incorporates Byte-Pair Encoding (BPE) to handle tokenization. Both models leverage the RNN's capacity for sequence processing, but their approaches to text representation differ significantly.

1) Model 1: Simple RNN with Word Tokenization

Model 1 is based on a simple RNN architecture that employs word tokenization to convert text data into numerical sequences. The input to this model is a sequence of integers representing words, which is first processed by an embedding layer. This layer transforms the integer sequences into dense vector representations, capturing semantic information about each word. The core of the model is a GRU layer, which effectively handles the sequential nature of language data by learning dependencies between words in the sequence. The GRU layer's output is then passed through a TimeDistributed dense layer with ReLU activation, which expands the feature space before the final Dense layer. This last layer, equipped with a softmax activation function, outputs probabilities for each word in the French vocabulary, enabling the model to generate translated sentences.

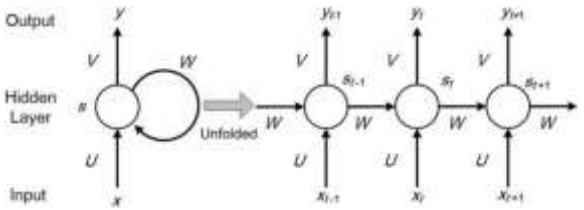

Fig.2. RNN Architecture

Fig. 2 illustrates the structure and unfolding mechanism of a Recurrent Neural Network (RNN), a type of neural network designed to process sequential data. On the left, the diagram shows the compact form of an RNN with three main components: the input (x), the hidden layer (s), and the output (y). The hidden layer has a looped connection, representing the way the network maintains memory by feeding the previous hidden state back into itself, allowing it to capture information from prior steps in the sequence. The weight matrices U, W, and V govern the input-to-hidden, hidden-to-hidden, and hidden-to-output transformations,

respectively. The right side of the figure demonstrates the unfolded version of the RNN over time. This "unfolding" shows how the network processes each element in a sequence by passing the hidden state from one time step to the next. For instance, at time step t, the network takes an input x_t, updates the hidden state s_t based on both the current input and the previous hidden state (s_t-1), and then produces an output y_t. This recurrent structure allows RNNs to process sequences of variable length and capture dependencies over time. It enables the network to "remember" earlier information, which is crucial for tasks like language modeling, where context from previous words is needed to predict the next word. The unfolded architecture highlights the network's ability to model time-series data, learning both short-term and long-term patterns across the sequence. This recurrent connection makes RNNs powerful for applications involving temporal dynamics, such as speech recognition, language translation, and sequential decision-making tasks. By maintaining internal states, RNNs can learn relationships between distant points in a sequence, giving them a distinct advantage over traditional feedforward networks.

The architecture of Model 1 is straightforward and provides a good baseline for sequence-to-sequence tasks. However, its reliance on word tokenization can lead to challenges with handling out-of-vocabulary words, especially when dealing with less frequent terms or novel words not seen during training. Despite these limitations, the model's simplicity makes it a solid choice for initial experimentation and offers insights into the basic workings of RNN-based translation systems.

2) Model 2: RNN with Embedding and Byte-Pair Encoding (BPE): Model 2 builds upon the RNN architecture by incorporating Byte-Pair Encoding (BPE) for improved tokenization. BPE addresses the limitations of word tokenization by merging frequent character pairs into subword units, which helps in managing out-of-vocabulary (OOV) issues. This approach allows the model to handle a broader range of text inputs, including those with unseen words, by representing them as combinations of known subword tokens. The model architecture mirrors that of Model 1 in terms of layers but starts with sequences tokenized using BPE. The embedding layer then processes these subword sequences into dense vectors, which are further refined by the GRU layer to capture the temporal relationships in the data.

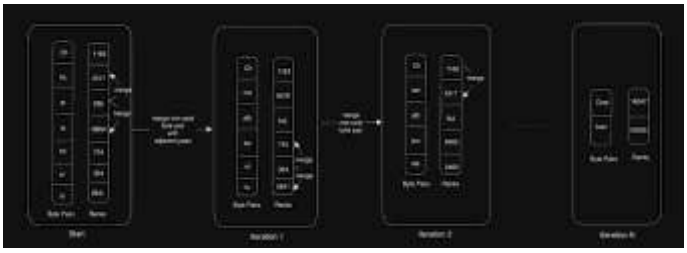

Fig.3. Byte-Pair Encoding

Fig. 3 illustrates the Byte Pair Encoding (BPE) algorithm, a tokenization and data compression technique widely used in natural language processing (NLP). Initially, byte pairs are assigned ranks based on their frequency of occurrence, as seen in the first column (labeled "Start"). For instance, the pairs "Ch", "ha", and "at" are associated with ranks 1163, 4317, and 266, respectively. During each iteration, the algorithm identifies and merges the byte pair with

the lowest rank—i.e., the most frequent pair. In Iteration 1, the pair "at" is merged with adjacent pairs, generating the new pair "hat" with an updated rank. This process continues iteratively, as in Iteration 2, where additional byte pairs such as "bot" and "ots" emerge from similar merging steps. The algorithm repeats this process, continually merging the most frequent pairs and updating ranks, leading to the creation of higher-level token structures as seen in the later iterations. By the final iteration (Iteration N), complex token units such as "Chat" and "bots" have emerged, with much higher ranks than in earlier iterations. This recursive merging of byte pairs captures frequent subword structures, making BPE an efficient mechanism for creating compact vocabularies in language models. Through this process, BPE enhances both token efficiency and model performance by segmenting text into common byte patterns that reduce out-of-vocabulary tokens during inference.

Following the GRU layer, the output is processed by a TimeDistributed dense layer with ReLU activation, enhancing the feature representation before the final Dense layer. This last layer, equipped with a softmax activation function, generates the probability distributions over the French vocabulary. The use of BPE in Model 2 allows for more flexible and robust handling of the translation task, particularly in dealing with rare or novel words, and demonstrates an advanced approach to tokenization in sequence-to-sequence models.

3) Model 3: Helsinki-NLP OPUS-MT: The Helsinki-NLP OPUS-MT English-French model represents a significant advancement in neural machine translation, built upon the Transformer architecture [1, 15]. This model, trained on the OPUS parallel corpus, implements a sequence-to-sequence framework [4] with multi-head attention mechanisms, making it particularly effective for English to French translation tasks. Unlike BERT's bidirectional approach [13], OPUS-MT utilizes an encoder-decoder architecture [2] specifically optimized for machine translation, where the encoder processes the source English text while the decoder generates the target French translation.

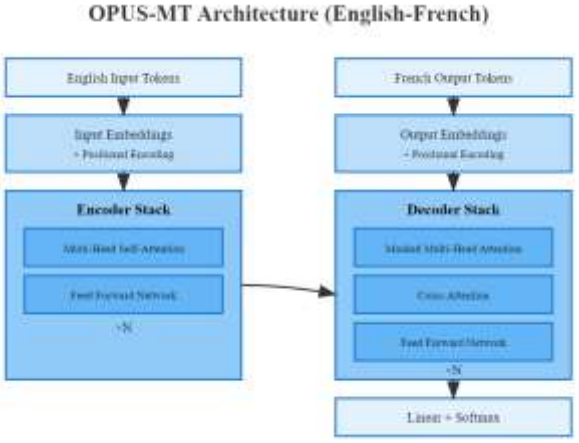

Fig.4 Architecture of Helsinki-NLP OPUS-MT

The model's architecture in Fig.4 incorporates advanced attention mechanisms [12] combined with subword tokenization [9, 10] to handle vocabulary variations between English

and French effectively. Building upon the success of neural machine translation systems [16], OPUS-MT employs positional encodings and layer normalization, similar to the standard Transformer architecture [1]. The model demonstrates particular strength in handling rare words [17] and maintaining contextual accuracy through its attention layers, making it suitable for both general-purpose translation and domain-specific applications. Its implementation through the Hugging Face platform makes it readily accessible for practical applications, following the democratization trend of neural machine translation tools [18].

F. Accuracy Comparison

In this section, the accuracy of both models is compared: the simple RNN using word tokenization and the RNN using Byte-Pair Encoding (BPE). The performance of each model was evaluated based on validation accuracy across multiple epochs. The results clearly indicate that the model utilizing BPE achieves higher accuracy than the word-tokenized model. This outcome is expectedas BPE allows for better handling of out-of-vocabulary words by breaking them down into smaller, more frequent subword units.

To visualize this comparison, the validation accuracy for both models is plotted over a series of epochs. The BPE model demonstrates consistently higher accuracy, making it a more suitable choice for tasks involving the translation of text with a diverse vocabulary. The graph clearly illustrates the superiority of BPE over simple word tokenization in terms of model accuracy.

Fig. 4 provides a comparative analysis of model accuracy over 15 epochs for two tokenization approaches: the Byte Pair Encoding (BPE) model and the Word Tokenization model. The blue curve, representing the BPE model, exhibits consistently higher accuracy throughout the training process, starting at approximately 0.95 accuracy at epoch 0 and peaking around 0.975. Although there is a minor fluctuation around epoch 10, the BPE model remains robust, maintaining high performance. In contrast, the green curve, representing the Word Tokenization model, shows significantly lower accuracy, beginning near 0.775 and gradually increasing to just above 0.80 by the final epoch. The word-level tokenization model demonstrates slower improvements and plateaus early, showing minimal gains beyond epoch 6. This figure highlights the superior performance of the BPE model, particularly in handling subword tokenization more efficiently, leading to more accurate and stable learning outcomes across epochs.

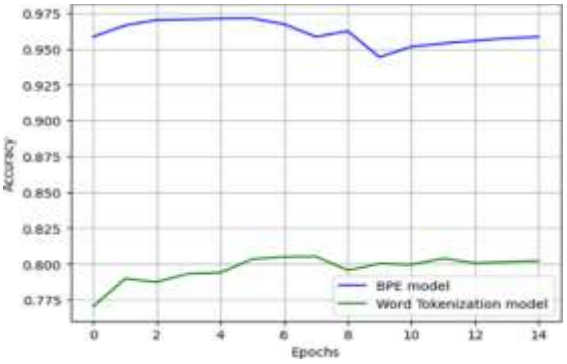

Fig.4. Accuracy

V. CONCLUSION

In this project, two Recurrent Neural Network (RNN) models for language translation were explored: one using simple word tokenization and another employing byte pair encoding (BPE) with an embedding layer. The RNN with BPE demonstrated a significant improvement over the word tokenized model, as BPE effectively managed subword units and handled larger vocabularies more efficiently. The RNN architecture allowed the models to capture the temporal dynamics of the input data, making them effective for sequence-to-sequence tasks, particularly for translating English sentences into French. The overall performance, as indicated by accuracy scores, showed that byte pair encoding played a crucial role in achieving better translation results.

Although the models showed promising outcomes, especially with BPE, there are still limitations that need addressing. The simple RNN architecture can struggle with long-range dependencies, which can affect the translation quality for longer and more complex sentences. Incorporating more advanced techniques, such as attention mechanisms, would allow the model to focus on specific parts of the input sequence, which could enhance performance. Another area to explore would be increasing the model's robustness to handle multiple languages or using larger, more diverse datasets to improve generalization.

Future work in this domain can go beyond RNNs and explore state-of-the-art models like transformers, which have shown superior results in various natural language processing tasks. Techniques such as self-attention and pre-training on large language corpora could further boost the model's ability to understand and translate complex language patterns. The development of more efficient, real-time language translation models could also lead to practical applications in various industries, including education, communication, and global commerce. This project serves as a steppingstone towards more sophisticated and capable translation systems.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," Advances in Neural Information Processing Systems, vol. 30, 2017.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," arXiv preprint arXiv:1406.1078, 2014.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," arXiv preprint arXiv:1409.0473, 2014.

[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," Advances in Neural Information Processing Systems, vol. 27, 2014.

[5] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, 1997.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[7] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," arXiv preprint arXiv:1602.02373, 2016.

[8] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network-based language model," in Interspeech, 2010, pp. 1045-1048.

[9] T. Kudo and J. Richardson, "SentencePiece: A simple and language-independent subword tokenizer and detokenizer for neural text processing," arXiv preprint arXiv:1808.06226, 2018.
[10] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.
[11] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135-146, 2017.
[12] M. T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015.
[13] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proceedings of NAACL-HLT, 2019, pp. 4171-4186.
[14] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.
[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," Advances in Neural Information Processing Systems, vol. 30, 2017.
[16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, and J. Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv preprint arXiv:1609.08144, 2016.
[17] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to Remember Rare Events," arXiv preprint arXiv:1703.03129, 2017.
[18] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "OpenNMT: Open-Source Toolkit for Neural Machine Translation," arXiv preprint arXiv:1701.02810, 2017.
[19] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI Technical Report, 2018.
[20] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato, "Unsupervised Machine Translation Using Monolingual Corpora Only," arXiv preprint arXiv:1711.00043, 2017.