Neural Network for Language Models and Its Applications

A Written Creative Work submitted to the faculty of
San Francisco State University
In partial fulfillment of
the requirements for
the Degree

Master of Science

In

Computer Science

by

Ashmitha Dale Pais

San Francisco, California

May 2024

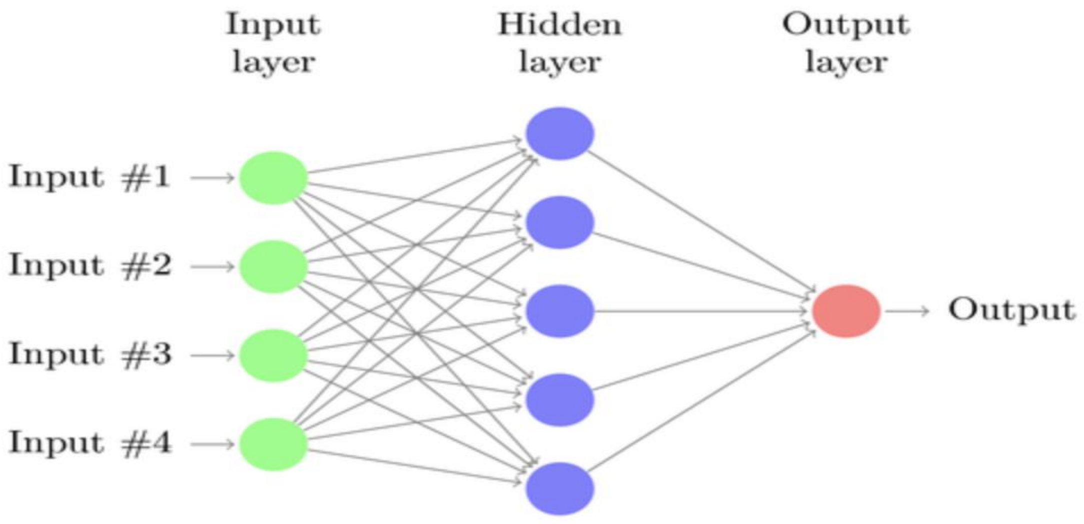
Table of Contents

List of Figures

Introduction

Research in language modeling for speech recognition has progressively shifted towards utilizing neural networks. Two contrasting approaches have emerged: Firstly, feedforward neural networks which adopt an n-gram methodology, and secondly, recurrent neural networks capable of learning contextual dependencies extending beyond a predetermined number of preceding words.

Figure 1: Neural Network Diagram



Besides analyzing the acoustic signal, modern speech recognition systems also leverage prior knowledge of human language structure. This knowledge is typically integrated through a probabilistic language model (LM), assigning probabilities to word sequences w1N. By assigning higher probabilities to word sequences more commonly found in natural language, speech recognition systems can identify the most likely spoken words.

State-of-the-art systems usually generate a set of hypotheses for the spoken word sequence using a backing-off language model. In a subsequent rescoring step, additional techniques are applied to refine the estimates for $p(w_1^N)$ on this limited set of hypotheses. The final recognition result is then obtained based on the updated language model probabilities.

Significant enhancements have been observed with the adoption of neural network language models (NNLMs) during rescoring. However, there are notable distinctions in how neural networks have been applied in speech recognition tasks previously.

When utilizing a feedforward neural network (FFNN), only the immediate (n - 1) preceding words are used to predict the probability of the current word. Although it's feasible to include words from the previous sentence, typically the history is truncated at the beginning of the sentence in the n-gram approach. Conversely, when employing a recurrent neural network, the entire sequence of preceding words is considered for predicting the word.

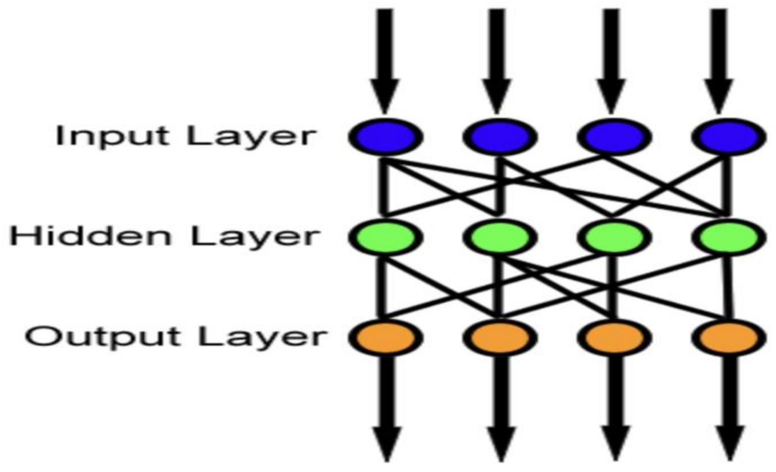
However, applying recurrent neural networks to rescoring standard word lattices is not feasible. Instead, only a subset of the hypotheses encoded in a word lattice can be considered and converted into a linear, non-branching format known as an m-best list.

Moreover, recurrent models cannot be consistently evaluated in rescoring scenarios. Speech recognizers only provide multiple hypotheses for a single sentence-like portion of the speech signal. To cover context lengths spanning, for example, two consecutive sentences, the hypotheses for the two sentences must be concatenated and rescored with the recurrent NNLM. However, for larger m-best lists and longer contexts, this approach becomes computationally infeasible. Consequently, in practice, only the best scoring hypothesis of a sentence is used when rescoring the following sentence. Ultimately, perplexities computed with recurrent models may not fully reflect the corresponding performance in terms of word error rate (WER) because when computing perplexities, the issue of exponential growth of the hypothesis space doesn't arise, and the recurrent model can be evaluated precisely.[4]

Comparison of different Neural Network Language Models

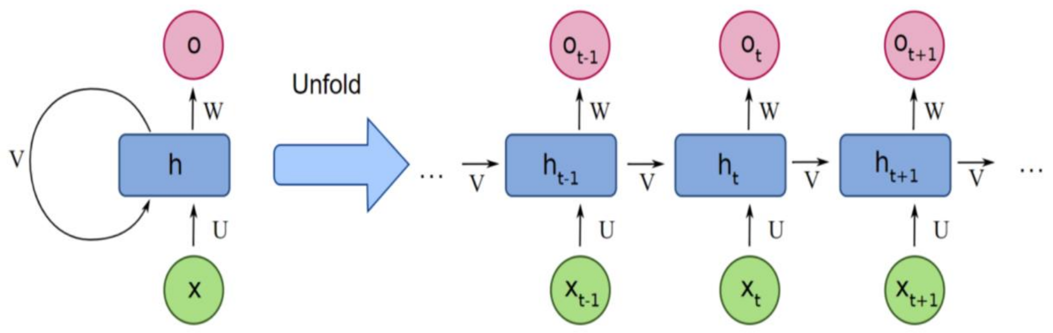
In the realm of neural network language models, various architectures exist, each with its unique characteristics and applications. Among these, Feedforward Neural Networks (FNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks stand out as prominent choices.

Figure 2: Feedforward Neural Network Diagram



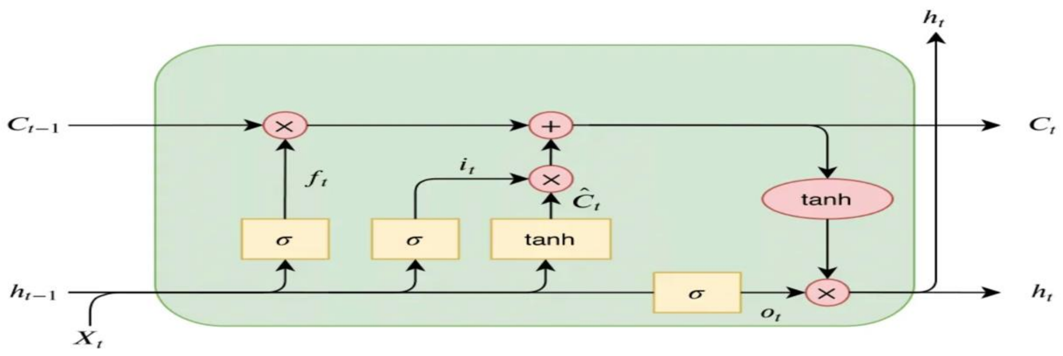
Feedforward Neural Networks (FNNs) constitute a basic yet powerful architecture in neural network language modeling. In FNNs, information flows unidirectionally from input to output through one or more hidden layers. This architecture is adept at capturing local dependencies within sequences but may struggle with long-range dependencies due to the lack of recurrent connections.

Figure 3: Recurrent Neural Network Diagram



Conversely, Recurrent Neural Networks (RNNs) excel in modeling sequential data by maintaining a hidden state that evolves over time. This recurrent connection allows RNNs to capture dependencies over arbitrary time steps, making them well-suited for tasks requiring context understanding, such as language modeling. However, traditional RNNs may suffer from the vanishing or exploding gradient problem, hindering their ability to capture long-term dependencies effectively.

Figure 4: Long Short-Term Memory Network Diagram



To address the limitations of traditional RNNs, Long Short-Term Memory (LSTM) networks were introduced. LSTMs incorporate gated mechanisms to regulate the flow of information through the network, enabling better long-term memory retention. The key components of LSTMs, including input, forget, and output gates, facilitate the selective retention and propagation of information, making them particularly effective for tasks involving extensive context modeling, such as language modeling on large datasets.

In summary, while Feedforward Neural Networks offer simplicity and efficiency, they may struggle with capturing long-range dependencies. Recurrent Neural Networks overcome this limitation by maintaining a hidden state, enabling them to model sequential data effectively. However, traditional RNNs may encounter issues with vanishing or exploding gradients. Long Short-Term Memory networks address these challenges by incorporating gated mechanisms, making them well-suited for tasks requiring extensive context understanding, such as language modeling on large datasets.

Article Contributions

The research paper [3] does a comparison on Recurrent and Feedforward Neural Networks. In Feedforward Neural Networks (FNN), the word w's class, denoted as c(w), is predicted, followed by the derivation of the probability of the next word given c(w) and the preceding words u, v. Since a limited number of words occur within a given class, and the number of word classes can be significantly smaller than the vocabulary size, considerable training speed-ups can be achieved by employing this equation.

$$p(w|u,v) = p\left(w|c(w);u,v\right) \cdot p\left(c(w)|u,v\right)$$

Recurrent neural networks share a similar structure, but they differ in that they only receive the immediately preceding word as input. The hidden layer incorporates recurrent connections, implicitly considering multiple preceding words presented to the network previously.

Training any type of Neural Network Language Model (NNLM) incurs significant computational costs. Therefore, training computations are distributed among multiple CPU cores, each independently generating intermediate results. In the case of feedforward NNLMs, a fixed number of n-grams from the training data are propagated through the network without weight updates. The number of training samples is referred to as the bunch size, with reasonable values typically ranging from 2 to 128.

However, this approach is not directly applicable to recurrent NNLMs due to the absence of a fixed context length, resulting in interdependencies among subsequent words in a sentence. Nevertheless, a similar concept can be applied to recurrent neural networks. Instead of parallelizing over individual words from a sentence, the bunch can be defined to encompass entire sentences.

Consequently, each core calculates gradients for a single sentence while keeping weights fixed. Subsequently, gradients for individual sentences are accumulated, and the resulting gradient is used to update weight parameters.

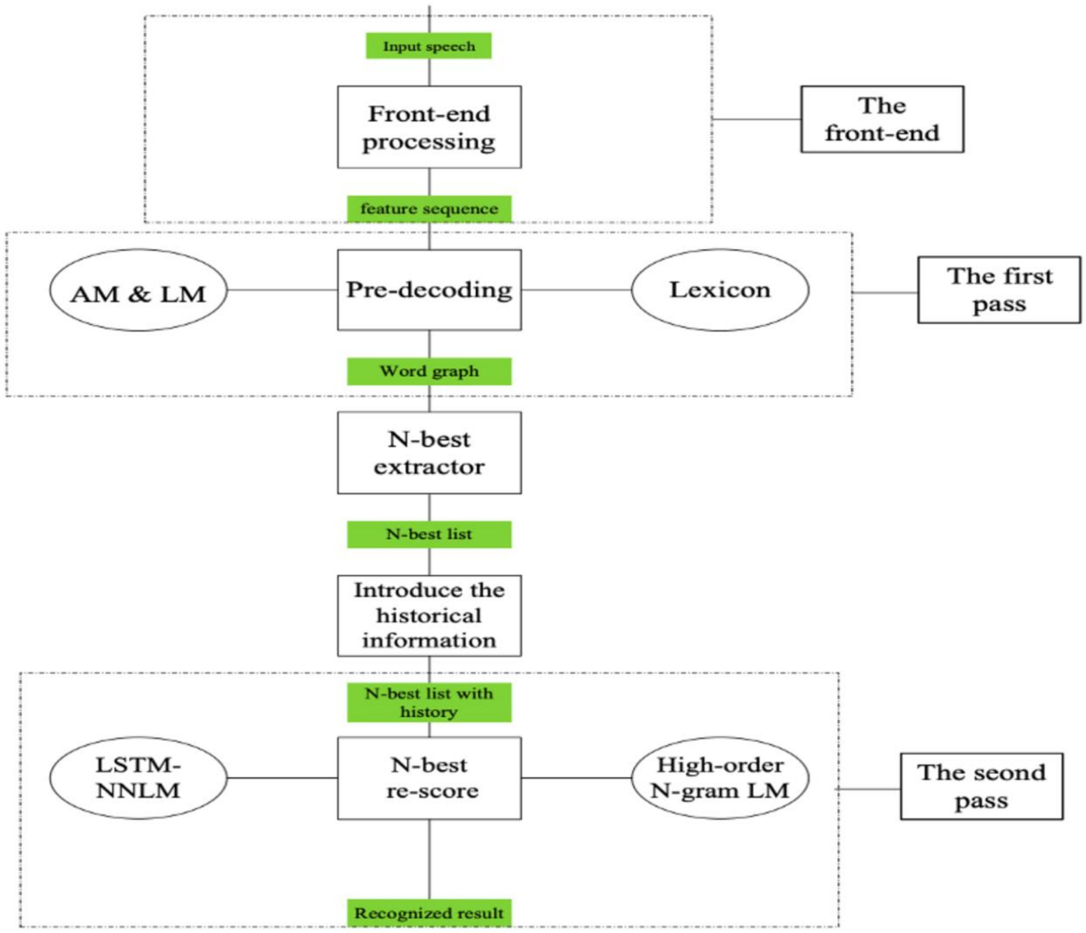
The conventional approach involves each CPU core maintaining its own gradient vector, with gradients being added up one by one (or pair-wise in a recursive manner if this step is parallelized). However, a primary challenge of bunch mode parallelization lies in its interaction with clustering speed-up techniques. The fundamental idea of clustering is to utilize only a small portion of the weight matrix for probability calculation, thereby necessitating computation of only a small number of gradient components. Considering the full gradient matrix would be inefficient.

To address this issue, the research tracks all word classes observed within a given training sequence. When updating weight parameters, only weights corresponding to words belonging to previously observed classes need modification. However, weights related to the class posterior probability p(c(w)|h), given the preceding history words h, must be updated in all cases.

By employing four CPU cores, they achieved a training time reduction to 60% of the original single-threaded version and found that Recurrent Neural Networks outperformed Feedforward Neural Networks.

Research Paper [4] talks about Neural Network Language Models in Speech Recognition. There exist prominent examples of speech recognition systems in practical settings, such as the voice assistants Siri and Alexa, both utilizing the RNN model. As illustrated in Figure 5, a conversational speech recognition system comprises front-end processing, involving operations on the original speech, including feature extraction, followed by sending the detected voice data to subsequent steps. The feature sequence undergoes processing in the pre-decoding module, leading to the generation of a word lattice by the decoding module, from which n-best lists are typically extracted. In the second pass, NNLM and a recurrent neural LM are employed to re-score the hypotheses in the n-best lists. Ultimately, the hypothesis with the highest new score is output as the final recognized result.

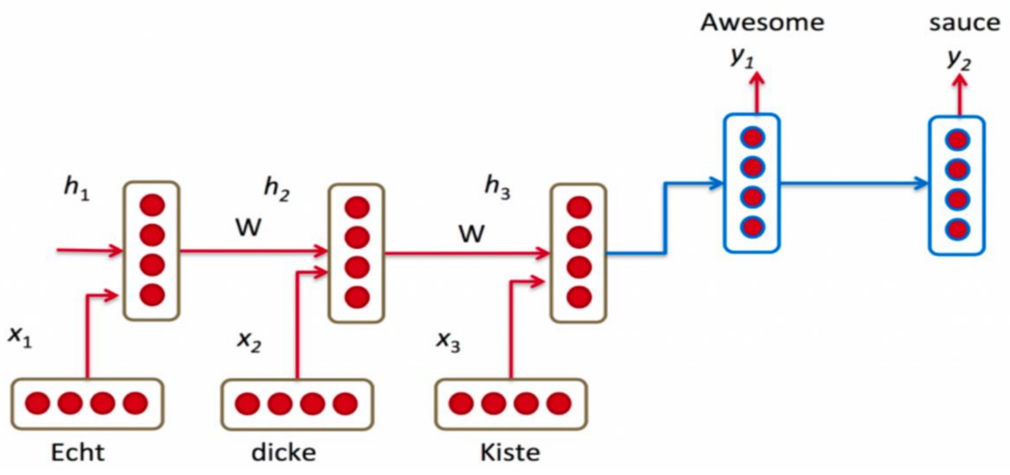
Figure 5: Conversational Telephone Speech Recognition System



While the RNN architecture inherently leverages long-range historical information, it has been observed that standard gradient-based training algorithms struggle to learn RNN weight parameters effectively to exploit such dependencies. This challenge arises because as dependencies extend over longer ranges, the gradient calculation becomes increasingly unstable. To tackle this issue, the LSTM structure is introduced in the hidden layer, effectively addressing the vanishing gradient problem.

Research Paper [2] talks about Neural Networks in Machine Translation. Some well-known examples of machine translation systems include Google Translate and Microsoft Translator, both of which employ the RNN model. The advancement of statistical machine translation technology is rapidly progressing, with numerous new models and methods emerging. These developments yield excellent results in translating simple or fixed sentences with specific applications. In a study by [7], a method is proposed that enhances the recurrent neural network language model by incorporating word vector characteristics to measure discourse coherence and adapt to various tasks more effectively. This approach expands the feature layer within the input layer. During model training, the enhanced structure integrates a context word vector via the feature layer, enhancing the model's ability to learn long-distance information dependencies.

Figure 6: Machine Translation of a German Phrase



For instance, consider the translation of a German phrase to an English sentence depicted in Figure 6. Most Neural Machine Translation (NMT) systems encode the source sentence (e.g., a German sentence) into a vector using a Recurrent Neural Network, and then decode an English sentence based on that vector, also using an RNN. In the illustration, the words "Echt," "Dicke," and "Kiste" are fed into an encoder, and after a special signal (not shown), the decoder initiates the production of a translated sentence. The decoder proceeds to generate words until a special end-of-sentence token is produced. In this scenario, the h vectors represent the encoder's internal state.

Research Paper [5] talks about strategies for training large scale neural networks. The article outlines an effective approach for training neural network-based language models on extensive datasets. Notably, faster convergence and improved overall performance when training data are organized based on their relevance is observed. Additionally, a hash-based implementation of a maximum entropy model, which can be integrated into the neural network model is presented. This integration notably reduces computational complexity. In the experiments, they attained approximately a 10% relative reduction in word error rate on an English Broadcast News speech recognition task compared to a large 4-gram model trained on 400 million tokens.

Training an RNN model with direct connections has shown promising results in terms of both perplexity and word error rate, even when employing very small hidden layers. For instance, the RNNME-40 model, comprising only 40 neurons, achieved nearly equivalent performance to the RNN-320 model, which utilizes 320 neurons. They've demonstrated that the direct connections in the NN model can be viewed as a maximum entropy model, emphasizing the importance of jointly training the RNN and ME models. Essentially, by employing this novel RNN architecture, we can reduce training times from weeks to days. Additional experiments conducted on a Wall Street Journal setup, previously utilized, showcased significant improvements. For instance, the word error rate reduced from 17.2% to 14.9% using the RNNME-100 model, trained in a single day on 36 million training tokens.

Research paper [1] talks about how large scale neural network training can be done efficiently. During the forward pass, neural networks retain the activations necessary for subsequent backpropagation, which can sometimes lead to a significant memory overhead, rendering training impractical. To mitigate this issue, two primary approaches are commonly employed: rematerialization (also known as checkpointing) and offloading.

F_i/B_i denotes the computation of the forward/backward pass on module i; F_{in} represents an operation that computes the output of F_i and discards its input; F_{ic} calculates the output of F_i and retains the input; F_{ie} calculates the output of F_i, retains the input, and preserves all intermediate activations necessary for later computation of B_i. These distinct operations can be readily implemented in frameworks such as PyTorch or TensorFlow.

Rematerialization involves a strategy wherein only a portion of the activations is stored during the forward pass, while the remaining activations are recomputed during the backward pass. Different rematerialization methods can be categorized based on the computational graphs they handle.

The first group originates from Automatic Differentiation (AD) and focuses on finding optimal schedules for homogeneous sequential networks. These networks consist of layers executed sequentially, each with identical computational and memory costs.

The second group addresses transition models, such as heterogeneous sequential networks. These networks encompass various sequential neural network architectures, such as CNNs, ResNet, and some transformers. Methods in this group adapt solutions from AD to accommodate heterogeneous settings.

The final group concentrates on general graphs. However, solving this problem optimally becomes NP-complete in the strong sense, necessitating solutions via Integer Linear Programming (ILP). Otherwise, approximate solutions can be obtained using various heuristics.

Offloading (also called Memory Swapping) is a technique that saves GPU memory by offloading activations to CPU memory during forward pass and prefetching them back into GPU memory for the corresponding backward computation.

Critical Study

In [1], it was demonstrated that employing very small hidden layers in a training RNN model with direct connections can yield commendable performance in terms of perplexity and word error rate. For instance, the RNNME-40 model, comprising only 40 neurons, achieved performance nearly on par with the RNN-320 model, which utilizes 320 neurons. The authors noted that the direct connections in the NN model could be regarded as a maximum entropy model and emphasized the importance of concurrently training the RNN and ME models. This novel RNN architecture significantly reduced training times from weeks to days. Additional experiments conducted on a Wall Street Journal setup [8] showcased notable improvements, such as reducing the error rate from 17.2% to 14.9% using an RNNME-100 model trained in a single day on 36 million training tokens.

However, it's worth exploring whether training the RNN model alongside a more complex maximum entropy model, such as model M, could be advantageous. [5] observed that while recurrent NNLMs offer potential benefits, their effectiveness is somewhat constrained by the limitations of ordinary speech recognition technology. The transition from n-gram to LSTM language models has steadily reduced error rates. Furthermore, re-scoring in telephone conversation speech recognition tasks using recognized results from the first pass decoding has been found beneficial, particularly when LSTM is utilized in hidden layers for recurrent neural networks, as it consistently yields the best results with the lowest error rates.

Moreover, [7] presents an improved recurrent neural network language model-based approach, modeling sentence-scaled language models while incorporating recurrent neural networks into discourse coherence modeling. During training, the model employs common methods of language models and dynamically adjusts the learning ratio based on the training effect of each round, thus expediting model training, and facilitating optimal solution discovery. Experimental results demonstrate that the recurrent neural network-based sentence-scaled language model notably improves discourse coherence modeling compared to optimal entropy discourse coherence.

Despite these advancements, recent attention has shifted towards more advanced neural network models, such as the Transformer, proposed to address machine translation challenges. The Transformer relies heavily on the attention mechanism to capture dependencies and employs a self-attention mechanism in each step, modeling relationships between all words in a sentence regardless of their positions. Unlike recurrent neural networks, the Transformer can process input sequences in parallel, leveraging GPU efficiency and accelerating training times. Furthermore, its multi-headed attention layer effectively mitigates the vanishing gradient problem, making it a promising choice for machine translation systems.

Conclusion

From the comprehensive analysis of multiple articles, it becomes apparent that Recurrent Neural Networks (RNNs) hold a significant advantage over standard feedforward approaches in speech recognition systems. The intrinsic feedback loop in RNNs facilitates the recycling of data back into the input, allowing for iterative refinement and processing, ultimately leading to improved performance and output accuracy. Additionally, integrating Long Short-Term Memory (LSTM) units into recurrent neural networks further enhances model capabilities and contributes to a reduction in error rates across various speech recognition tasks.

Expanding on this, recurrent neural network-based sentence-scaled language models have emerged as a preferred choice for machine translation systems due to their ability to capture contextual information effectively. These models have consistently demonstrated promising results compared to other neural network architectures.

However, as technology evolves, there's growing anticipation surrounding the adoption of transformer neural network models in machine translation systems. Transformers, with their attention mechanisms and parallel processing capabilities, offer the potential for significant advancements and superior performance over recurrent neural networks. By leveraging transformer models, researchers and practitioners aim to achieve breakthroughs in machine translation accuracy and efficiency.

Therefore, while recurrent neural networks have proven their efficacy, the ongoing pursuit of innovation and exploration of transformer models signifies an exciting frontier in the field of machine translation and natural language processing.

References

[1]. Mikolov, T., Deoras, A., Povey, D., Burget, L. & Cernocky, J. Strategies for training large scale neural network language models. In Proc. Automatic Speech Recognition and Understanding 196–201 (2011).

[2]. Julia Gusak, Daria Cherniuk, Alena Shilova, Alexander Katrutsa, Daniel Bershatsky, Xunyi Zhao, Lionel Eyraud-Dubois, Oleg Shlyazhko, Denis Dimitrov, Ivan Oseledets, Olivier Beaumont " Survey on Large Scale Neural Network Training " https://arxiv.org/abs/2202.10435

[3]. C. Xue, "A Discourse Parser Language Model Based on Improved Neural Network in Machine Translation," 2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), 2018, pp. 605-608, doi: 10.1109/ICITBS.2018.00158.

[4]. M. Sundermeyer, I. Oparin, J. -L. Gauvain, B. Freiberg, R. Schlüter and H. Ney, "Comparison of feedforward and recurrent neural network language models," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8430-8434, doi: 10.1109/ICASSP.2013.6639310.

[5]. L. Zuo, X. Wan and J. Liu, "Comparison of Various Neural Network Language Models in Speech Recognition," 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), 2016, pp. 894-898, doi: 10.1109/ICISCE.2016.195.

[6]. Bengio, Y, Simard, P, Frasconi, P, Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5, 1994: 157–166

[7]. C. Xue, "A Discourse Parser Language Model Based on Improved Neural Network in Machine Translation," 2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), 2018, pp. 605-608, doi: 10.1109/ICITBS.2018.00158.

[8]. T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Cernock´ ˇ y, "Empirical evaluation and combination of advanced language modeling techniques," in Proceedings of Interspeech, 2011.

[9]. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin "Attention Is All You Need". Submitted on 12 Jun 2017 at arXiv.org.

[10]. Chuang-Hua Chueh, Jen-Tzung Chien and Hsin-min Wang, "A maximum entropy approach for integrating semantic information in statistical language models," 2004 International Symposium on Chinese Spoken Language Processing, 2004, pp. 309-312, doi: 10.1109/CHINSL.2004.1409648.