Paper on Recurrent Neural Networks (RNNs): Evolution of RNN to ChatGPT transformer

By
Name: Choy Jee Hung Caleb
Student ID: 2341475
Class: DAAA/FT/2B/23

In partial fulfillment of the requirements for the module
ST1504 DEEP LEARNING
CA2 Technical Paper
Lecturer: Dr Willson

Submission Date: 05/08/2024

1. Abstract

This paper delves into the evolution of Recurrent Neural Networks (RNNs) and their pivotal role in Natural Language Processing (NLP). The paper examines the architecture and workings of large language models (LLMs) like ChatGPT, a state-of-the-art language model based on the Transformer architecture and its training methodologies. We will also explore its design, advantages over traditional RNNs, and its effectiveness in natural language understanding and generation tasks.

2. Introduction

Natural Language Processing (NLP) has become an integral part of modern technology, enabling machines to understand, interpret, and generate human language. From machine translation to sentiment analysis, NLP applications are diverse and impactful. Among the various techniques used in NLP, Recurrent Neural Networks (RNNs) have emerged as a powerful tool for processing sequential data, such as text.

Basic RNNs are designed to handle sequences by maintaining a hidden state that captures information from previous time steps. However, they struggle with capturing long-term dependencies due to issues like vanishing gradients. To overcome these limitations, advanced architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) have been developed. These architectures incorporate mechanisms to retain information over extended periods, addressing the shortcomings of basic RNNs.

Beyond these improvements, the advent of large language models, particularly those based on the Transformer architecture, has revolutionized the field. Models like GPT-3, developed by OpenAI, demonstrate remarkable capabilities in understanding and generating human-like text. This paper delves into these advancements, with a focus on ChatGPT, a variant of GPT-3, to explore how their architecture surpasses traditional RNNs in NLP tasks.

3. Related Works

3.1. Natural Language Processing (NLP)

NLP is a field of artificial intelligence that focuses on the interaction between computers and human languages. The goal of NLP is to enable computers to internalize, read, and replicate human language that is both meaningful and useful. Among the various techniques used in NLP, Recurrent Neural Networks (RNNs) have emerged as a powerful tool for processing sequential data, such as text.

3.2. Recurrent Neural Networks (RNNs)

RNNs have been a cornerstone in NLP, effectively capturing sequential dependencies in text data. Basic RNNs are designed to handle sequences by maintaining a hidden state that captures information from previous time steps (Elman, 1990). However, they struggle with capturing long-term dependencies due to issues like vanishing gradients (Bengio et al., 1994). To overcome these limitations, advanced architectures like Long Short-Term Memory (LSTM) were developed.

3.3. Long Short-Term Memory (LSTM)

LSTMs were designed to overcome the challenges of learning long-range dependencies, a limitation inherent in traditional RNNs (Hochreiter & Schmidhuber, 1997). LSTMs use a gated mechanism, consisting of input, forget, and output gates, which regulates the flow of information. This design allows LSTMs to retain information over longer periods, making them highly effective for complex tasks in natural language processing and speech recognition. However, LSTMs struggle with processing very long sequences, as the time and computational resources required scale linearly with the sequence length. LSTMs also face challenges with vanishing gradients, although lesser than RNNs. This is especially experienced in very deep networks. This makes them less practical for some real-world applications that involve massive datasets or require rapid processing.

3.4. Gated recurrent units (GRU)

GRUs, proposed by Cho et al. (2014), offer a simplified version of LSTMs with fewer parameters, maintaining similar performance while being computationally efficient. This makes them a good choice for NLP and other areas where processing sequential data is important. The performance on music, speech and NLP was found to be closely equal to that of LSTM. The ability to balance performance with computational efficiency made them a valuable tool in the field of deep learning (Bai et al., 2018), especially in scenarios where resources are limited or when faster training times are desired. However, like LSTM, they still struggle with long-term dependencies due to the vanishing gradient problem.

3.5. Transformers

The advancement in AI has led to the development of a more powerful architecture, the Transformer introduced by Vaswani et al. (2017) from the scientific paper “Attention is all you need”.. Instead of processing data sequentially, transformers use a mechanism known as 'self-attention' to process entire sequences of data in parallel. The attention mechanism is introduced in the paper by Bahdanau et al. (2014) to solve the long input problem, especially with encoder-decoder type RNNs. The attention mechanism works by assigning a weight to each input in the sequence. These weights affect the amount of attention the model should pay to each input when producing an output. This enables the model to capture complex and distant relationships in the data more effectively. Transformers eliminate the need for recurrence entirely, leading to significant improvements in training efficiency and scalability. This architecture is used in many models, including OpenAI’s GPT (Brown et al., 2020) and Google’s BERT (Devlin et al., 2019).

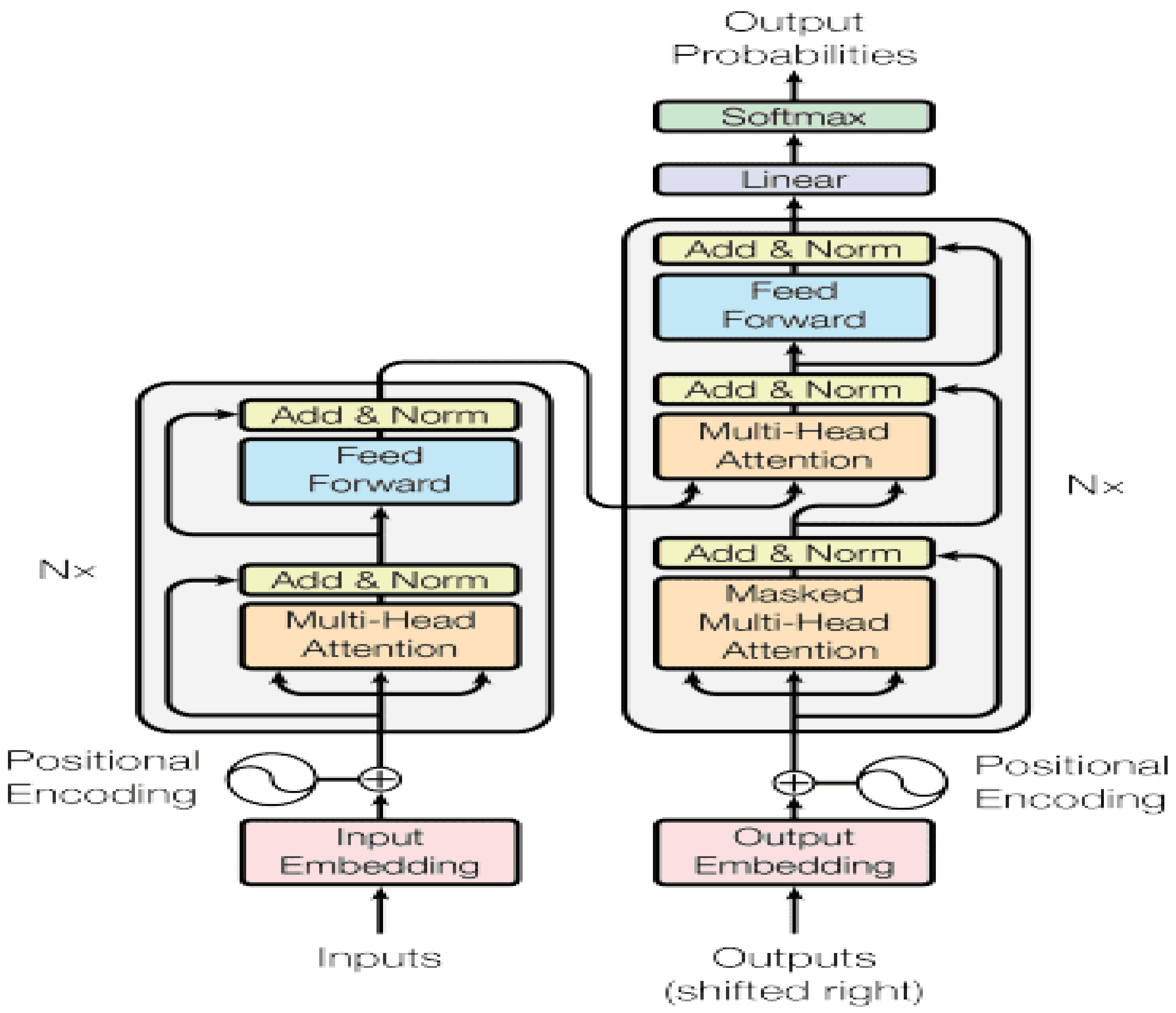

Figure 1: Transformers Architecture (Vaswani *et al.* 2017)

3.5.1. How transformers work

At their core, transformers leverage a self-attention mechanism that allows each token in a sequence to weigh and consider every other token, capturing intricate relationships and contextual information. They utilize positional encodings to retain the sequence order, as they process the entire input simultaneously rather than sequentially. Multi-head attention enhances this by enabling the model to focus on different parts of the sequence in parallel. The architecture consists of encoders and decoders: encoders build contextual representations of the input, while decoders generate the output based on these representations. This approach, combined with layer normalization and residual connections, significantly improves training stability and efficiency.

3.6. Bidirectional Encoder Representations from Transformers (BERT)

BERT (Bidirectional Encoder Representations from Transformers) is a model introduced by Devlin et al. (2019) that uses the Transformer architecture for natural language understanding. Unlike previous models that processed text in a unidirectional manner, BERT processes text bidirectionally, meaning it considers both the left and right context of a word simultaneously. This bidirectional approach allows BERT to capture more nuanced meanings and relationships between words (Devlin et al., 2019). BERT is pre-trained on a large corpus of text using two primary tasks: the Masked Language Model (MLM) and Next Sentence Prediction (NSP). In MLM, some words in the input text are masked, and the model learns to predict these masked words based on their context. In NSP, the model learns to predict whether one sentence follows another in the given text, which helps it understand sentence relationships (Devlin et al., 2019). The pre-trained BERT model can be fine-tuned for specific tasks such as text classification, question answering, and named entity recognition by adding task-specific layers on top of the pre-trained model. This transfer learning approach allows BERT to achieve state-of-the-art performance on various natural language processing benchmarks (Devlin et al., 2019).

3.7. Generative Pre-training Transformer (GPT)

ChatGPT is based on the Transformer architecture introduced by Vaswani et al. (2017). This architecture uses self-attention mechanisms to process and generate text. The training of ChatGPT involves both supervised and reinforcement learning. Initially, the model is trained using unsupervised learning on a vast corpus of text data. It is then fine-tuned with supervised learning, where human reviewers provide feedback to refine the model's responses (Radford et al., 2021). Text input to ChatGPT is tokenized into sub words or word pieces, which are converted into vectors. These vectors are processed through the model's layers to generate predictions, a method detailed by Devlin et al. (2019) in their work on BERT (Devlin et al., 2019). ChatGPT generates responses based on the context of the input, using attention mechanisms to focus on different parts of the text. This allows the model to produce coherent and contextually relevant output (Brown et al., 2020). For text generation, techniques such as beam search and sampling are employed to maximize the likelihood of the next token (Holtzman et al., 2020). ChatGPT uses the scaled Dot-Product Attention. This calculates the score by using the dot product of the query and key, and the result is scaled by the square root of the dimension of the key. Another way is the Additive Attention where the score is counted by adding the query and key together after which a nonlinear function is then applied. While more computationally expensive, it can sometimes yield better results.

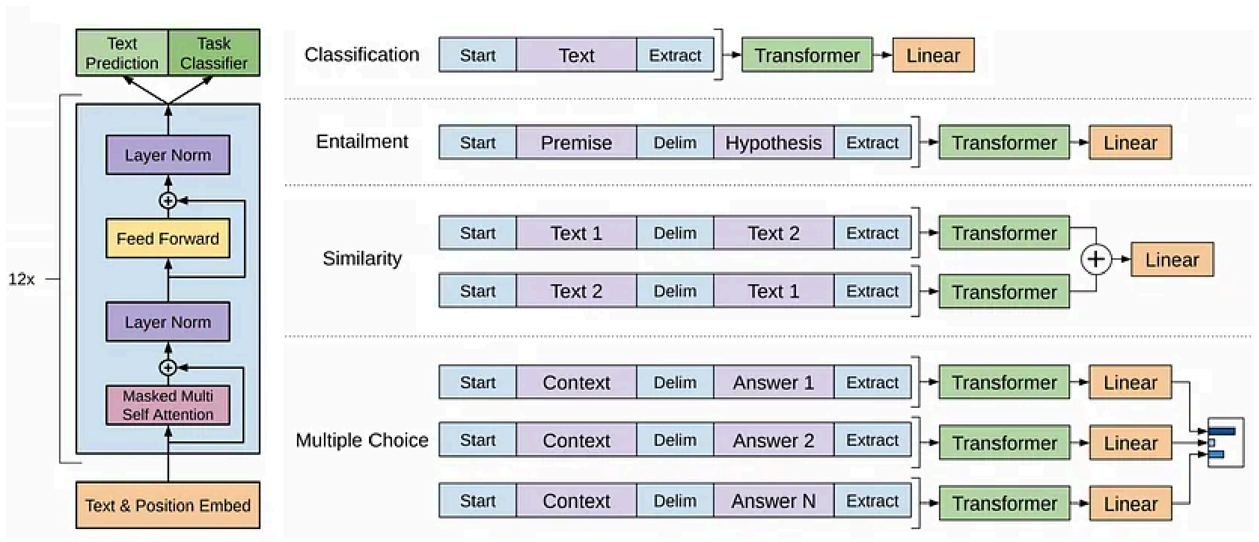

Figure 3: ChatGPT Architecture

4. Dataset/Methodology/Experiment

4.1. Dataset

The Penn Treebank (PTB) corpus, a widely used benchmark dataset in NLP, serves as the basis for our experiments. The PTB corpus consists of annotated English text from the Wall Street Journal, providing a robust dataset for evaluating language models. The dataset is split into training, validation, and test sets, ensuring a standardized evaluation framework.

4.2. Methodology

We implement and compare the performance of four RNN using architectures: RNN, LSTM, GRU and Transformer. Each model is trained on the PTB dataset to predict the next word in a sequence, a fundamental task in language modeling. Afterwhich, a metric known as perplexity is returned.

4.2.1. Test perplexity

Perplexity is a metric used to evaluate the performance of probabilistic models, particularly in the context of language models. It measures how well a probability distribution or a model predicts a sample. In simpler terms, perplexity can be understood as the inverse probability of the test set, normalized by the number of words. Lower perplexity indicates that the model is better at predicting the sample.

4.3. Experiment

The training involves optimizing the models using the Adam optimizer with a learning rate of 0.001. The models are tuned to an extent and trained for 50 epochs, with early stopping based on the validation loss to prevent overfitting. The performance is evaluated using perplexity, a standard metric for language models that measures the model's ability to predict the test set. The code to this experiment can be found in the Jupyter notebook provided with this paper.

5. Discussion

Model (Earliest to Latest)	Expected Perplexity Score obtained from papers on PTB	Perplexity Score obtained from my models on PTB
RNNs (1986)	103.50 (Diao, 2019)	105.44
GRU (2014)	92.48 (Bai et al., 2018)	104.82
LSTM (1997)	78.93 (Bai et al., 2018)	103.80
Transformer (2017)	20.50 (Brown et al., 2020)	23.35

Table 1: Perplexity Score of different RNNs

Although the way the models are built from the papers that provided the expected scores are different from the models I built due to the difference in parameters such as epochs, batch size etc, the concept of each architecture remains the same. What we are really looking at the expected score is for the trend of decrease in perplexity score, which shows the models of RNNs evolution over time.

The results demonstrate the superiority of advanced RNN architectures over basic RNNs. LSTMs and GRUs significantly reduce the perplexity compared to basic RNNs, highlighting their enhanced ability to capture long-term dependencies in sequential data. The gating mechanisms in LSTMs and GRUs enable the models to retain relevant information over extended periods, addressing the vanishing gradient problem effectively. GRU would have a slightly higher perplexity score due to having a simpler architecture than LSTM however the difference would not be too great since GRU was built off LSTM.

ChatGPT, based on the Transformer architecture, further improves upon these results. The self-attention mechanisms in ChatGPT allow it to handle long-range dependencies more effectively than traditional RNN-based models. Fine-tuning transformer through adding in learning rate scheduler, further improves the perplexity score allowing for better generation and prediction of text.

6. Conclusion

This paper explores the advancements in RNN architectures and their development in NLP. While basic RNNs laid the foundation for sequential data processing, advanced models like LSTMs and GRUs have significantly improved performance by addressing the limitations of their predecessors. The emergence of the transformer architecture, such as ChatGPT, marks a significant leap in NLP capabilities, enabling more sophisticated and human-like text generation.

The practical experiments conducted on the PTB dataset validate the superior performance of advanced RNN models over basic RNNs. The evolution of RNNs models from the past to the present demonstrates the real-world potential of large language models in conversational AI in the future.

References

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5, 157-166. https://doi.org/10.1109/72.279181

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14, 179-211. https://doi.org/10.1207/s15516709cog1402_1

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998-6008). https://doi.org/10.48550/arXiv.1706.03762

Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate. https://doi.org/10.48550/arXiv.1409.0473

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. https://doi.org/10.48550/arXiv.2005.14165

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. https://doi.org/10.48550/arXiv.1810.04805

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.48550/arXiv.2103.00020

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. https://doi.org/10.48550/arXiv.1810.04805

Holtzman, A., Buys, J., Forbes, M., & Choi, Y. (2020). The curious case of neural text degeneration. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. https://www.cscjournals.org/library/manuscriptinfo.php?mc=IJCL-136#MCAI

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. https://doi.org/10.48550/arXiv.1803.01271

Diao, E., Ding, J., & Tarokh, V. (2019). Restricted recurrent neural networks. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 56-63). https://doi.org/10.1109/BigData47090.2019.9006257