Eva Lucero Pérez Salcedo A01568830
7th Semester
Artificial Emotions:Going beyond Artificial Intelligence
Instituto Tecnologico de Monterrey campus Guadalajara
Course provided by: Mahdi Zaarei and Alejandro de León Languré
20th September 2023

LSTM vs GRU: Comparison performance and architecture

Abstract

In this paper we will cover certain subjects about two types of recurrent neural networks, LSTM and GRU. Describing it's general use and performance as well as it's most significant differences. The objective of this research is to understand its implications in different areas of use as well as the important details that conform each of these RNNs. In terms of memory cell structure, LSTMs offer a more complex design with separate memory cells, facilitating selective memory retention and forgetting over extended sequences. In contrast, GRUs, with their streamlined architecture, its more for a balance between memory and computational efficiency.

Introduction

RNN (Recurrent neural networks) are tools in data analysis, using their sequential methods to generate sequences and understanding it's purpose. RNNs as it were traditionally made had various difficulties with the vanishing gradient problem, unable to capture such long-range dependencies in data, meaning that this LRD couldn't sustain long memory processes. For this reason, more advanced architectures were made such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) where this specific problem was solved. LSTM and GRU had a great impact in deep learning, overcoming this issue traditional RNNs had. These type of advanced architectures stand out in tasks where it involves sequential data, NLP, speech recognition and time-series predictions. In the next sections, a general view in these architectures are provided such as it's structures on the memory cell, number of gates, information retention, computational efficiency and parameterization to understand the core principles behind these methods and how they contributed to the advancement of artificial intelligence.

History of LSTM and its structure

LSTMs were introduced by Sepp Hochreiter and Jürgen Schmidhuber in a paper titled "Long Short-Term Memory" published in 1997. As mentioned previously, the objective which it was created was to solve the problem of the vanishing gradient in traditional RNNs and improve their ability to capture and retain long-term dependencies in sequential data, the problem with RNNs in training were the gradients of the loss function that tended to decrease, causing this to have difficulties in learning and retaining information from distant past time steps. Hochreiter and Schmidhuber architecture made it so this issue were handled with LSTM, through the use of memory cells and gating mechanisms. [1]
The key innovation of LSTMs lies in their ability to selectively update and erase information in the memory cell, allowing the network to learn what information to remember or forget over extended sequences. The architecture includes three gates listed below. [2]

Input Gate (i_t - Input):

- The input gate determines which information from the current input should be stored in the memory cell.
- It is controlled by a sigmoid activation function, which outputs values between 0 and 1.
- Mathematically, the input gate operation can be expressed as:
  $i_t = \sigma(W_{ii}^{i} x_t + b_{ii} + W_{hi} h_t - 1 + b_{hi})$
  where i_t is the input gate output, x_t is the current input, $h_t - 1$ is the previous hidden state,σ is the sigmoid activation function, and $Wii,\ bii,\ Whi$, and b_{hi} are the weights and biases associated with the input gate.

Forget Gate (f_t - Forget):

- The forget gate decides what information from the previous state should be discarded or forgotten.
- Like the input gate, it is controlled by a sigmoid activation function.
- Mathematically, the forget gate operation is given by:
  $f_t = \sigma(W_{if} x_t + b_{if} + W_{hf} h_{t-1} + b_{hf})$
  where f_t is the forget gate output, x_t is the current input, h_{t-1} is the previous hidden state and σ is the sigmmoid activation function.

Output Gate (o_t - Output):

- The output gate determines what the next hidden state h_t and the output y_t should be based on the current input and the memory cell content.

- The output gate uses both a sigmoid and a $tahn$ (hyperbolic tangent) activation function.
- Mathematically, the output gate operation is expressed as:
  $o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$
  $g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$
  $h_t = o_t \cdot g_t$
  where o_t is the output gate output, g_t is the candidate new memory content with a multiplication.

Each one of these enabling the model to learn when to read, reset and exit the information processed.

> *Recurrent networks … have an internal state that can represent context information. … [they] keep information about past inputs for an amount of time that is not fixed a priori, but rather depends on its weights and on the input data.*
>
> *…*
>
> *A recurrent network whose inputs are not fixed but rather constitute an input sequence can be used to transform an input sequence into an output sequence while taking into account contextual information in a flexible way.*

— Yoshua Bengio, et al., Learning Long-Term Dependencies with Gradient Descent is Difficult, 1994. [3]

History of GRU and its structure

Introduced by Cho,et al. in 2014, the Gated Recurrent Unit also has the purpose to solve the vanishing gradient problem in the RNNs. Its also considered a variation of LSTM because of their similar design. In this case GRU uses two gates, update and reset. It can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

The architecture includes two gates listed below. [4]

Update gate (z_t for time step t):

- We start with calculating the z_t for time step t using the formula:
  $z_t = \sigma(W^{(Z)}x_t + U^{(z)}h_{t-1})$

- When x_t is plugged into the network unit, it is multiplied by its own weight $W_{(z)}$. The same goes for $h_{(t-1)}$ which holds the information for the previous t-1 units and is multiplied by its own weight $U_{(z)}$. Both results are added together and a sigmoid activation function is applied to squash the result between 0 and 1.
- The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future. That is really powerful because the model can decide to copy all the information from the past and eliminate the risk of vanishing gradient problem.

Reset gate:

- Essentially, this gate is used from the model to decide how much of the past information to forget. To calculate it, we use:
- $r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$
- This formula is the same as the one for the update gate. The difference comes in the weights and the gate's usage.

Table Comparison

	LSTM	GRU
Memory Cell Structure	LSTM has a more complex structure with separate memory cells and a set of gating mechanisms (input, output, and forget gates). The memory cell allows the network to store information for long periods.	GRU has a simpler structure with a single hidden state. It combines the memory cell and the hidden state, which reduces the complexity compared to LSTM
Number of Gates	LSTM has three gates—input gate, forget gate, and output gate. Each gate has a specific role in controlling the flow of information.	GRU has two gates—reset gate and update gate. The reset gate controls what information to discard from the previous state, and the update gate controls what

		new information to store in the current state.
Information Retention	The explicit memory cell in LSTM allows it to selectively remember or forget information over long sequences, which can be useful for tasks that require capturing long-term dependencies.	GRU has a more streamlined architecture and may be computationally more efficient. However, it might not perform as well as LSTM in tasks that require modeling long-term dependencies.
Computational Efficiency	LSTM is generally considered to be more computationally intensive due to its complex structure and the presence of multiple gates.	GRU is simpler and, therefore, might be computationally more efficient compared to LSTM. This simplicity can lead to faster training times in some cases.
Parameterization	LSTM has more parameters than GRU due to its additional gating mechanisms and the separate memory cell.	GRU has fewer parameters, which can be an advantage in scenarios where memory and computational resources are limited.
Erase of Training	LSTMs can be more robust for tasks that involve capturing long-term dependencies, but they may also be more prone to overfitting.	GRUs are simpler and may be easier to train on smaller datasets, but they might not perform as well as LSTMs on tasks requiring sophisticated memory management.

Results

The comparison between Long Short-Term Memory and Gaated Recurrent Unit has been researched and has been a large topic of discussion between the Artificial Intelligence

advancements. As it has been seen in the previous subsections, this analysis is based on the research keeping in mind their advantages and disadvantages depending on the task in question.
Both of these methods have shown a high performance in handling sequential data, including in applications with NLP, and speech recognition. LSTM, with their memory cells and multiple gating mechanisms, are highly recommended in capturing and retaining long-term dependencies. GRUs even tho they have a much simpler architecture, have more computational efficiency.
In memory cell structure, LSTM has a more complex structure with separate memory cells, it has a selective memory retention and forgetting over extended sequences while GRUs with a streamlined architecture, sacrifice a part of their memory to gain computational efficiency, depending on the task a type must be chosen in this case. Researchers and practitioners have continued to refine and adapt the LSTM architecture, and it remains an important component in neural network models.

Conclusion

In conclusion, LSTM and GRU architectures depends on the demands of the specific task, resources and data set. LSTM is specially good in memory control and capturing long-term dependencies but their computational effort might not be worth in some cases. While GRUs are more simple in a way that their efficiency relies on the computational subject.
As the neural network models continue to evolve, deciding between LSTM and GRU remains in considering the effectiveness of the sequential data analysis in hand. As we speak, researchers and people interested in neural networks, are developing more architectures like these to be refined and of use in future advancements regarding artificial intelligence.

References

[1] H. A. Fallahgoul. “Long-Range Dependency.” ScienceDirect. Accessed: Sep. 23, 2023. [Online]. Available: https://www.sciencedirect.com/topics/mathematics/long-range-dependency

[2]“A Gentle Introduction to Long Short-Term Memory Networks by the Experts - MachineLearningMastery.com.” MachineLearningMastery.com. Accessed: Oct. 27, 2023. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[3]S. Hochreiter. “LONG SHORT-TERM MEMORY.” Institute of Bioinformatics - Homepage. Accessed: Sep. 23, 2023. [Online]. Available: https://www.bioinf.jku.at/publications/older/2604.pdf

[4]S. Kostadinov. “Understanding GRU Networks.” Medium. Accessed: Sep. 23, 2023. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[5]“10.2. Gated Recurrent Units (GRU) — Dive into Deep Learning 1.0.3 documentation.” Dive into Deep Learning — Dive into Deep Learning 1.0.3 documentation. Accessed: Sep. 23, 2023. [Online]. Available: https://d2l.ai/chapter_recurrent-modern/gru.html

[6]R. Fu, Z. Zhang, and L. Li. “Using LSTM and GRU neural network methods for traffic flow prediction.” ResearchGate. Accessed: Sep. 23, 2023. [Online]. Available: https://www.researchgate.net/publication/312402649_Using_LSTM_and_GRU_neural_network_methods_for_traffic_flow_prediction