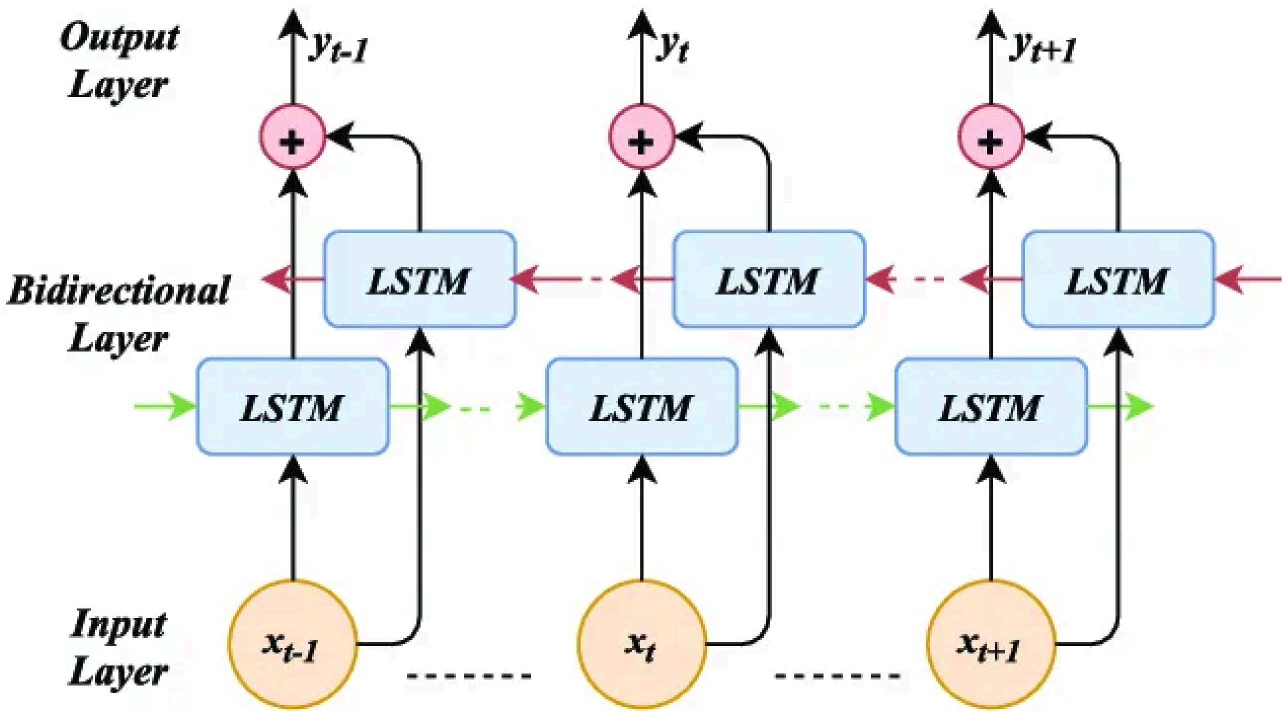
Understanding Bidirectional LSTM for Sequential Data Processing

6 min read · May 18, 2023

Anishnama Follow

Listen Share



Source

What is Bi-LSTM and How it works?

Bi-LSTM (Bidirectional Long Short-Term Memory) is a type of recurrent neural network (RNN) that processes sequential data in both forward and backward directions. It combines the power of LSTM with bidirectional processing, allowing the model to capture both past and future context of the input sequence.

To understand Bi-LSTM, let’s break down its components and functionality:

1. **LSTM (Long Short-Term Memory):** LSTM is a type of RNN designed to overcome the limitations of traditional RNNs in capturing long-term dependencies in sequential data. It introduces memory cells and gating mechanisms to selectively retain and forget information over time. LSTMs have an internal memory state that can store information for long durations, allowing them to capture dependencies that may span across many time steps.

2. **Bidirectional Processing:** Unlike traditional RNNs that process input sequences in only one direction (either forward or backward), Bi-LSTM processes the sequence in both directions simultaneously. It consists of two LSTM layers: one processing the sequence in the forward direction and the other in the backward direction. Each layer maintains its own hidden states and memory cells.

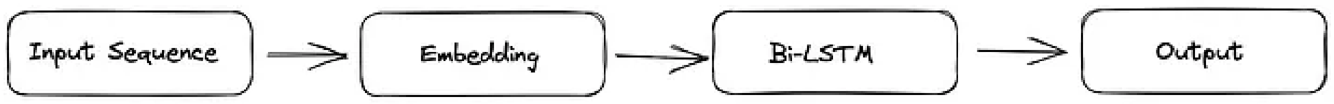
3. **Forward Pass:** During the forward pass, the input sequence is fed into the forward LSTM layer from the first time step to the last. At each time step, the forward LSTM computes its hidden state and updates its memory cell based on the current input and the previous hidden state and memory cell.

4. **Backward Pass:** Simultaneously, the input sequence is also fed into the backward LSTM layer in reverse order, from the last time step to the first. Similar to the forward pass, the backward LSTM computes its hidden state and updates its memory cell based on the current input and the previous hidden state and memory cell.

5. **Combining Forward and Backward States:** Once the forward and backward passes are complete, the hidden states from both LSTM layers are combined at each time step. This combination can be as simple as concatenating the hidden states or applying some other transformation.

The benefit of Bi-LSTM is that it captures not only the context that comes before a specific time step (as in traditional RNNs) but also the context that follows. By considering both past and future information, Bi-LSTM can capture richer dependencies in the input sequence.

Architecture



Let's break down each component of the architecture:

1. **Input Sequence:** The input sequence is a sequence of data points, such as words in a sentence or characters in a text. Each data point is typically represented as a vector or embedded representation.
2. **Embedding:** The input sequence is often transformed into dense vector representations called embeddings. Embeddings capture the semantic meaning of the data points and provide a more compact and meaningful representation for the subsequent layers.
3. **Bi-LSTM:** The Bi-LSTM layer is the core component of the architecture. It consists of two LSTM layers: one processing the input sequence in the forward direction and the other in the backward direction. Each LSTM layer has its own set of parameters.
4. **Output:** The output of the Bi-LSTM layer is the combination of the hidden states from both the forward and backward LSTM layers at each time step. The specific combination method can vary, such as concatenating the hidden states or applying a different transformation.

The Bi-LSTM layer processes the input sequence in both forward and backward directions simultaneously. During the forward pass, the LSTM layer captures information from the past (previous time steps), while during the backward pass, it captures information from the future (following time steps). This bidirectional processing allows the model to effectively capture long-term dependencies in the input sequence.

The output of the Bi-LSTM layer can be used for various purposes depending on the specific task. For example, in text classification, the output may be passed through a fully connected layer followed by a softmax activation to obtain class probabilities.

In sequence labeling tasks like named entity recognition, the output may be directly used to predict the label for each input token.

The architecture of a Bi-LSTM can be further extended or modified based on the specific requirements of the task. Additional layers, such as fully connected layers or attention mechanisms, can be added on top of the Bi-LSTM layer to further enhance the model's capabilities and performance.

Python Implementation

Here's an example of a Python implementation of a Bi-LSTM using the Keras library:

```
from keras.models import Sequential
from keras.layers import Embedding, Bidirectional, LSTM, Dense

# Define the model architecture
model = Sequential()

# Add an embedding layer to convert input sequences to dense vectors
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_lengt

# Add a Bidirectional LSTM layer
model.add(Bidirectional(LSTM(units=lstm_units, return_sequences=True)))

# Add a dense output layer
model.add(Dense(units=num_classes, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accu

# Print the model summary
model.summary()
```

In this code, you'll need to replace `vocab_size` with the size of your vocabulary, `embedding_dim` with the desired dimensionality of the embedding space, `max_sequence_length` with the maximum length of your input sequences, `lstm_units` with the number of LSTM units, and `num_classes` with the number of classes in your classification task.

The model starts with an embedding layer that converts the input sequences into dense vectors. The Bidirectional LSTM layer processes the embedded sequences in both forward and backward directions. Finally, a dense output layer with softmax activation is added for classification tasks.

After defining the model, you can compile it by specifying the loss function, optimizer, and metrics. Then, you can train the model on your labeled data using the `fit()` function.

Keep in mind that this is a simplified example, and you may need to adapt it to your specific use case by adding additional layers, adjusting hyperparameters, and preprocessing your data accordingly.

Pros and Cons of using Bidirectional- LSTM

Here are some potential pros and cons of using a bidirectional LSTM for a particular task:

dependencies in sequential data by processing input sequences in both forward and backward directions. This makes them well-suited for tasks that require modeling context over a long period of time, such as speech recognition or natural language processing.

2. **Improved performance:** Bidirectional LSTMs often perform better than traditional LSTMs on tasks such as speech recognition, machine translation, and sentiment analysis.

3. **Flexible architecture:** The architecture of bidirectional LSTMs is flexible and can be customized by adding additional layers, such as convolutional or attention layers, to improve performance.

Cons:

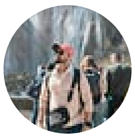
1. **High computational cost:** Bidirectional LSTMs can be computationally expensive due to the need to process the input sequence in both directions. This can make them impractical for use in resource-constrained environments.

2. **Requires large amounts of data:** Bidirectional LSTMs require large amounts of training data to learn meaningful representations of the input sequence. Without sufficient training data, the model may overfit to the training set or fail to generalize to new data.

3. **Difficult to interpret:** Bidirectional LSTMs are often seen as "black boxes," making it difficult to interpret how the model is making predictions. This can be problematic in applications where interpretability is important, such as medical diagnosis or financial analysis.

Overall, the decision to use a bidirectional LSTM should depend on the specific task at hand and the available resources, as well as the trade-offs between performance and interpretability.

Rnn · Bidirectional Lstm · Bidirectional Rnn · Deep Learning · TensorFlow



Follow

Written by Anishnama

138 followers · 5 following

Gen AI Expert, Data Scientist, Blogger, Writer, Trainer

Responses (4)

Write a response

What are your thoughts?

Ali Raza
Apr 30

impressive

Reply

Nitehrt
Feb 25

Excellent. Quick Ques: WHere in the python code do I specify the activation function for each time step which takes the sum of the output of the forward and backward LSTM to output each Y value? Thanks in advance!

Reply

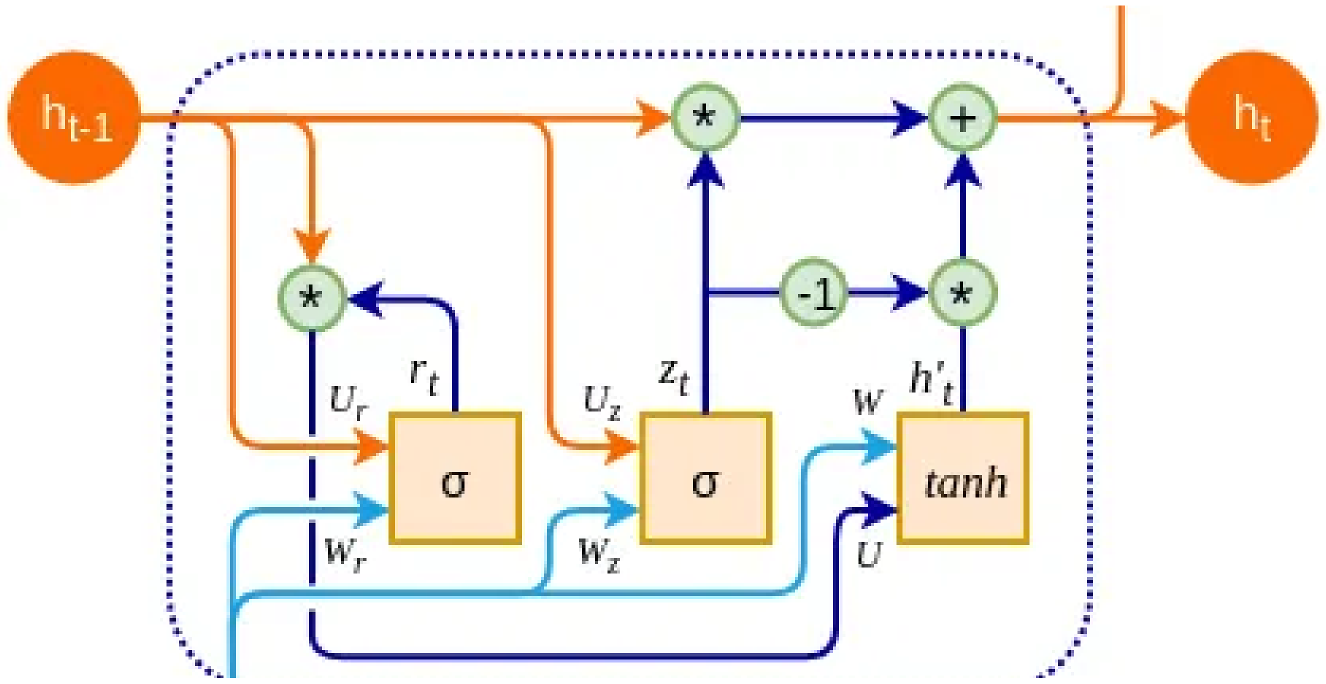
Ahnaf Al Islam
Oct 12, 2023

Hi there, could you perhaps recommend any books/research papers that discusses more about LSTM being better than other ML architectures for sequential data.

1 reply Reply

See all responses

More from Anishnama

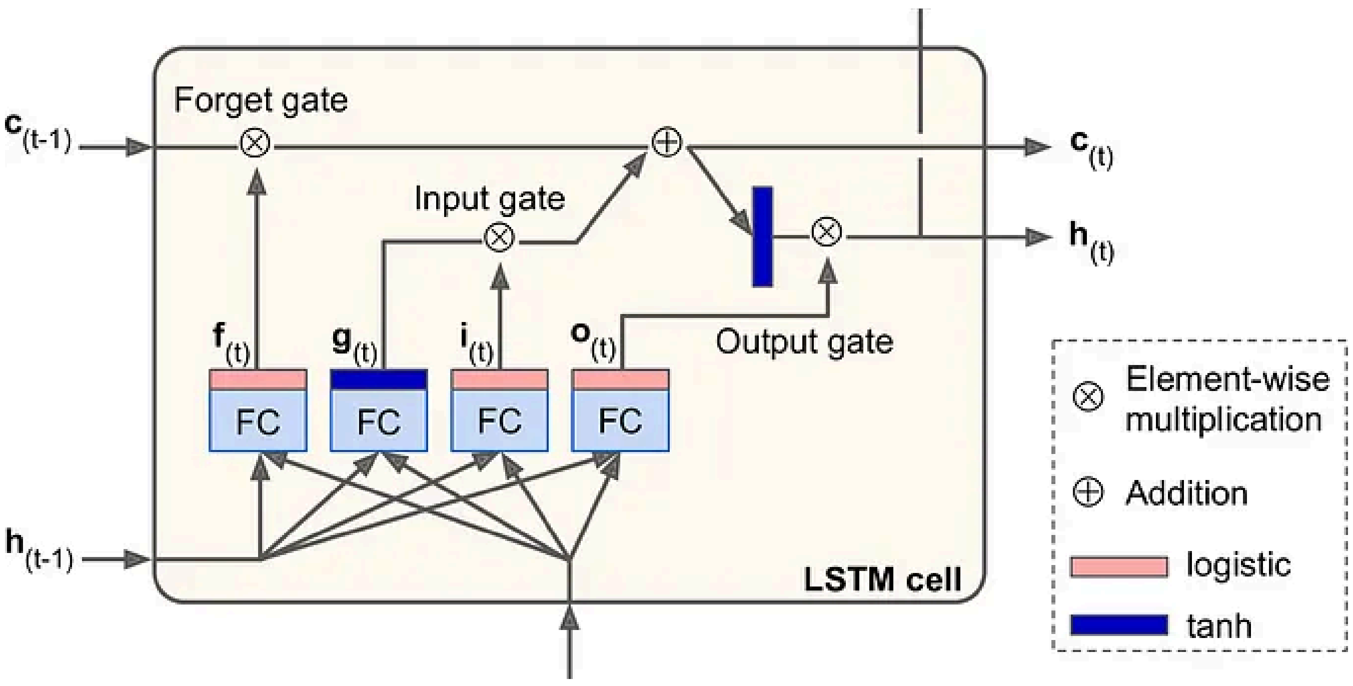


Anishnama

Understanding Gated Recurrent Unit (GRU) in Deep Learning

What is Gated Recurrent Unit(GRU) ?

May 4, 2023 350 5



Anishnama

Understanding LSTM: Architecture, Pros and Cons, and Implementation

What is LSTM and How it works?

Apr 28, 2023 276 5

Anishnama

Understanding Cost Functions in Machine Learning: Types and Applications

What is cost function?

Mar 29, 2023 200 3

Anishnama

Matthews Correlation Coefficient(MCC) one of the best metric when 2 classes are imbalanced

The Matthews correlation coefficient (MCC) is a measure of the quality of binary (two-class) classifications, which ranges from -1 to +1. A...

Jan 16, 2023 120

See all from Anishnama

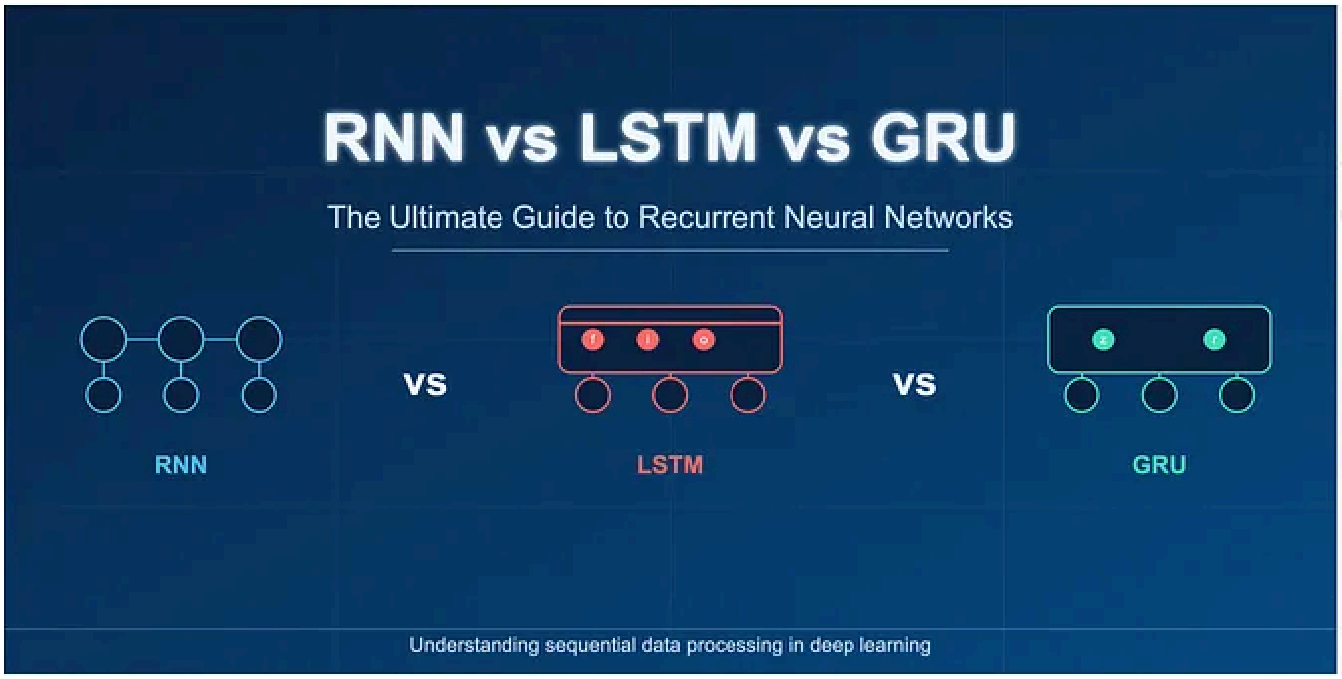
Recommended from Medium

Sanjay Singh

Deep Learning Neural Networks Explained: ANN, CNN, RNN, and Transformers (Basic Understanding)

Deep Learning is at the heart of modern Artificial Intelligence. From image recognition to language translation, neural networks power…

Aug 17 5


RNN vs LSTM vs GRU
The Ultimate Guide to Recurrent Neural Networks
RNN
vs
LSTM
vs
GRU
Understanding sequential data processing in deep learning

Saif Ali

RNN vs. LSTM vs. GRU

The Ultimate Guide to Recurrent Neural Networks

Apr 10 5

In Data Curiosity by Jeffrey Lee

From Words to Numbers: Understanding Word2Vec

All my articles are 100% free to read. Non-members can read for free by clicking my friend link!

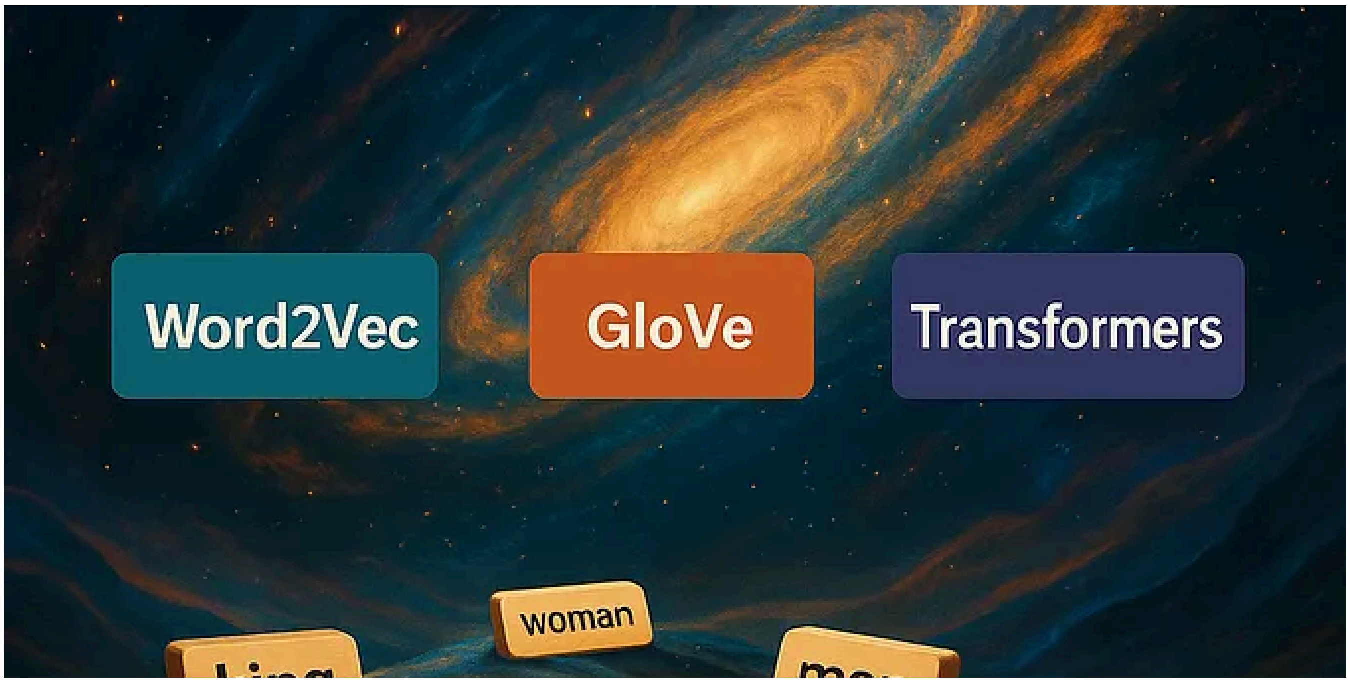
Mar 23 100

In Data Science Collective by Sunghyun Ahn

Transfer Learning: Why We Freeze and Unfreeze Model Layers

Choosing the Right Layers to Freeze When Fine-tuning | Data Science for Machine Learning Series (3)

Apr 30 106 3



Seema Nair

Word2Vec, GloVe, and Transformers: A Journey Through Word Embeddings (with Code!)

In the world of Natural Language Processing (NLP), how we represent words fundamentally shapes how machines understand language. From early...

Jul 8

Rezowanur Rahman Robin

LSTM RNN: An In-depth Look at its Architecture

1. Introduction: The Limitations of Traditional RNNs

Aug 10

See more recommendations