Analytics Vidhya

Tutorial on RNN | LSTM: With Implementation

BEGINNER DATA SCIENCE DATASETS NLP

Introduction

In NLP we have seen some NLP tasks using traditional neural networks, like text classification, sentiment analysis, and we did it with satisfactory results. but this wasn't enough, we faced certain problems with traditional neural networks as given below.

> - The prediction was highly dependent on the specific words.
> - If you change any word with it synonyms result gets deviated.3. If you write *like*, *affection* instead of *love*, the result gets deviated.4. The prediction was not dependent on the sequence of words.

To fix this problem we came up with the idea of Word Embedding and a model which can store the sequence of the words and depending on the sequence it can generate results.

*ie "**I love playing Ball**" and "**I do not love playing Ball**"*

In both the sentence we see "love" hence our vanilla model can classify it as a positive sentiment but as you see its negative sentiment, here sequence of words matters a lot. So far we had no model which can consider the sequence of words in a sentence for the prediction.

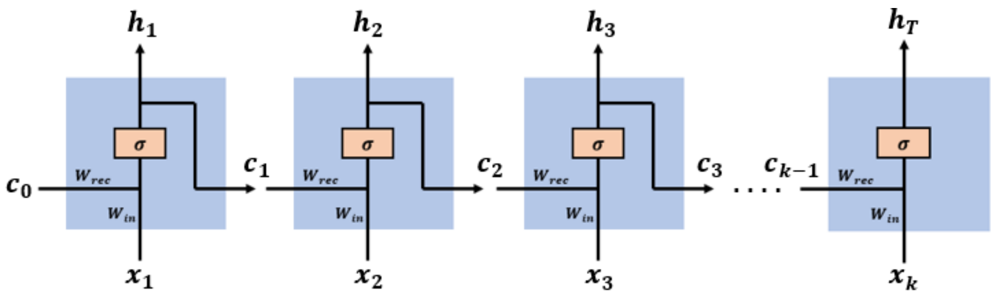
Table of contents

Learning Objectives

1. Understand the concept of **Recurrent Neural Networks (RNN)** and how they handle **sequential data**.
2. Learn how **Long Short-Term Memory** **(LSTM)** and **Gated Recurrent Units** solve the problem of learning long-term dependencies in sequential data.
3. Understand the role of **nodes** and **activation functions** in LSTM networks.

4. Step-by-step implementation of **LSTM networks** and understanding the role of the **loss function** in training these networks.
5. Learn about **Bidirectional LSTM (BI-LSTM) Networks** and how they process **input sequences**.
6. Implement LSTM, GRU, and BI-LSTM networks in a programming language.

RNN Model (Recurrent Neural Networks) Tutorial



Source: OpenSource

To solve this problem **Recurrent neural network** came into the picture. which solves this problem by using hidden layers. and hidden layers are the main features of a recurrent neural network. hidden layers help RNN to remember the sequence of words (data) and use the sequence pattern for the prediction.

The idea of RNN has developed a lot. it has been used for speech recognition and various NLP tasks where the sequence of words matters. RNN takes input as time series (sequence of words), we can say RNN acts like a memory that remembers the sequence.

Also learn **CNNS.**

Use cases of RNN

```
Speech, text recognition & sentiment classificationMusic synthesisImage captioningChatbots & NLP(natural language processing)Machine Translation — Language translationStock predictionsFinding and blocking abusive words in a speech or text
```

Does RNN sound so powerful? No ! there is a big problem.

Problems with RNN :

Exploding and vanishing gradient problems during backpropagation.

Gradients are those values which to update neural networks weights. In other words, we can say that Gradient carries information.

Vanishing gradient is a big problem in deep neural networks. it vanishes or explodes quickly in earlier layers and this makes RNN unable to hold information of longer sequence. and thus **RNN becomes short-term memory**.

If we apply RNN for a paragraph RNN may leave out necessary information due to gradient problems and not be able to carry information from the initial time step to later time steps.

To solve this problem LSTM, GRU came into the picture.

How do LSTM, GRU solve this problem?

I highly encourage you to read Colah's blog for in-depth knowledge of LSTM.

The reason for exploding gradient was the capturing of relevant and irrelevant information. a model which can decide what information from a paragraph and relevant and remember only relevant information and throw all the irrelevant information

This is achieved by using gates. the LSTM (Long -short-term memory) and GRU (Gated Recurrent Unit) have gates as an internal mechanism, which control what information to keep and what information to throw out. By doing this LSTM, GRU networks solve the exploding and vanishing gradient problem.

Almost each and every SOTA (state of the art) model based on **RNN** follows LSTM or GRU networks for prediction.

LSTMs /GRUs are implemented in speech recognition, text generation, caption generation, etc.

LSTM networks Step by Step

Every LSTM network basically contains three gates to control the flow of information and cells to hold information. The Cell States carries the information from initial to later time steps without getting vanished.

Gates

Gates make use of sigmoid activation or you can say *tanh* activation. values ranges in *tanh* activation are 0 -1.

1. Forget Gate
2. Input Gate
3. Output Gate

Let's see these gates in detail:

Forget Gate

This gate decides what information should be carried out forward or what information should be ignored.

Information from previous hidden states and the current state information passes through the sigmoid function. Values that come out from sigmoid are always between 0 and 1. if the value is closer to 1 means information should proceed forward and if value closer to 0 means information should be ignored.

Input Gate

After deciding the relevant information, the information goes to the input gate, Input gate passes the relevant information, and this leads to updating the cell states. simply saving updating the weight.

Input gate adds the new relevant information to the existing information by updating cell states.

Output Gate

After the information is passed through the input gate, now the output gate comes into play. Output gate generates the next hidden states. and cell states are carried over the next time step.

Gated Recurrent Unit

Gated Recurrent Units are similar to the LSTM networks. GRU is a kind of newer version of RNN. However, there are some differences between GRU and LSTM.

- GRU doesn’t contain a cell state
- GRU uses its hidden states to transport information

- It Contains only 2 gates(Reset and Update Gate)
- GRU is faster than LSTM
- GRU has lesser tensor's operation that makes it faster

Update Gate

Update Gate is a combination of Forget Gate and Input Gate. Forget gate decides what information to ignore and what information to add in memory.

Reset Gate

This Gate Resets the past information in order to get rid of gradient explosion. Reset Gate determines how much past information should be forgotten.

BI-LSTM Networks

We have seen how LSTM works and we noticed that it works in uni-direction.

Bidirectional long-short term memory networks are advancements of unidirectional LSTM. Bi-LSTM tries to capture information from both sides left to right and right to left. The rest of the concept in Bi-LSTM is the same as LSTM.

This improves the accuracy of models.

Implementation

We are going to perform a movie review (text classification) using BI-LSTM on the IMDB dataset. The goal is to read the review and predict if the user liked it or not.

Importing Libraries

```
import numpy as np from keras.models import Sequential from keras.preprocessing import sequence from keras.layers import Dropout from keras.layers import Dense, Embedding, LSTM, Bidirectional
```

Importing the dataset

Loading IMDB standard dataset using the Keras dataset class.

```
from keras.datasets import imdb (x_train, y_train),(x_test, y_test) = imdb.load_data(num_words=10000)
```

`num_words = 10000` signifies that only 10000 unique words will be taken for our dataset.

`x_train, x_test`: List of movie reviews text data. having an uneven length.

`y_train, y_test`: Lists of integer target labels (1 or 0).

Feature Extraction

Since we have text data in x_train and x_test of having an uneven length. Our goal is to convert this text data into a numerical form in order to feed it into the model.

Make the length of texts equal using padding.

We are defining `max_len = 200`. If a sentence is having a length greater than 200 it will be trimmed off otherwise it will be padded by 0.

```
max_len = 200 x_train = sequence.pad_sequences(x_train, maxlen=maxlen) x_test = sequence.pad_sequences(x_test, maxlen=maxlen) y_test = np.array(y_test) y_train = np.array(y_train)
```

Designing the Bi-directional LSTM

The way we will be defining our bidirectional LSTM will be the same for LSTM as well.

You can either use Sequential or Functional API to create the model. here we are using Sequential API.

```
model = Sequential() model.add(Embedding(n_unique_words, 128, input_length=maxlen)) model.add(Bidirectional(LSTM(64))) model.add(Dropout(0.5)) model.add(Dense(1, activation='sigmoid')) model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

- `input_length = maxlen` Since we have already made all sentences in our dataset have an equal length of 200 using pad_sequence.
- The Embedding layer takes `n_unique_words` as the size of the vocabulary in our dataset which we already declared as 10000.
- After the Embedding layer, we are adding Bi-directional LSTM units.
- Using sigmoid activation and then compiling the model

Training the model

We have prepared our dataset and model not calling the fit method to train our model.

```
history=model.fit(x_train, y_train, batch_size=batch_size, epochs=12, validation_data=[x_test, y_test])
print(history.history['loss']) print(history.history['accuracy'])
```

Result

The plotting result can tell us how effective our training was. Let’s plot the training results.

```
from matplotlib import pyplot pyplot.plot(history.history['loss']) pyplot.plot(history.history['accuracy'])
pyplot.title('model loss vs accuracy') pyplot.xlabel('epoch') pyplot.legend(['loss', 'accuracy'], loc='upper
right') pyplot.show()
```

As you can see the accuracy line is nearly touching the one and loss is minimum very close to zero. However, you can go ahead and draw some predictions using the model.

Conclusion

In this article, we learned about RNN, LSTM, GRU, BI-LSTM and their various components, how they work and what makes them keep an upper hand for NLP tasks. We saw the implementation of Bi-LSTM using the IMDB dataset which was ideal for the implementation didn’t need any preprocessing since it comes with the Keras dataset class.

Key Takeaways

1. **RNN Tutorial**: Gain a comprehensive understanding of Recurrent Neural Networks and their applications.
2. **LSTM and GRU**: Understand how LSTM and GRU solve the problem of learning long-term dependencies in sequential data.
3. **Step-by-Step LSTM**: Learn the step-by-step process of implementing LSTM networks, including the role of nodes, activation functions, and the loss function.
4. **BI-LSTM Networks**: Understand the concept and application of Bidirectional LSTM Networks in processing sequential data.
5. **Implementation**: Gain practical experience in implementing LSTM, GRU, and BI-LSTM networks using popular deep learning frameworks.

Frequently Asked Questions

Q1. How is LSTM different from RNN?

A. LSTM (Long Short-Term Memory) is a type of RNN (Recurrent Neural Network) that addresses the vanishing gradient problem of a standard RNN. LSTM introduces a **‘memory cell’** that can maintain information in memory for long periods of time. A standard RNN has difficulty in carrying information through many time steps (or ‘layers’) which makes learning long-term dependencies practically impossible. This is not an issue with LSTM due to its unique design.

Q2. Why does LSTM outperform RNN?

A. LSTM outperforms RNN as it can handle both short-term and long-term dependencies in a sequence due to its **'memory cell'**. This cell can keep important information throughout the processing of the sequence, and – via its **'gates'** – it can remove or diminish the information that is not relevant. This is particularly useful when dealing with **'current input'** that is affected by the distant past inputs in the sequence.

Q3. What is deep learning?

A. Deep learning is a subset of **machine learning**, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—to learn from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help optimize the results. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing tasks without human intervention.

Q4. What is a Gated Recurrence Unit (GRU)?

A. The Gated Recurrence Unit (GRU) is the newer generation of Recurrent Neural Networks and is pretty similar to an LSTM. GRU's got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate, which makes it simpler than LSTM and computationally more efficient.

Article Url - https://www.analyticsvidhya.com/blog/2022/01/tutorial-on-rnn-lstm-gru-with-implementation/

Abhishek Jaiswal