Bidirectional RNN (BiLSTM)

What is a Bidirectional RNN?

A Bidirectional RNN is like reading a sentence **both forwards and backwards** at the same time to better understand its meaning. It combines two separate RNNs (usually LSTMs or GRUs):

1. **Forward Layer**: Processes the sequence from start to end (like normal reading)
2. **Backward Layer**: Processes the sequence from end to start (reverse reading)

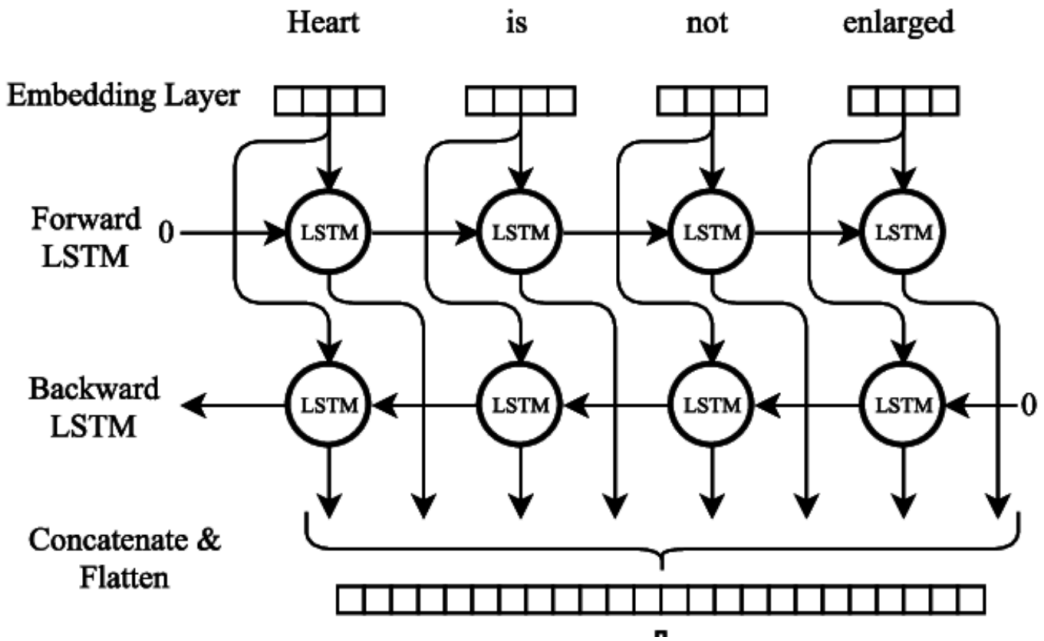
The outputs are combined at each time step, giving the model "context" from both past and future.

Sometimes, knowing the **future** helps understand the **present**.

For example:

- In a sentence:

  To understand the word *"bank"*, we might need **later words** to know if it means *"river bank"* or *"money bank"*.



✅ Solution: Bidirectional RNN

It runs **two RNNs in parallel**:

1. One processes from **left to right** (past → future)
2. One processes from **right to left** (future → past)

Then, it **combines both results**.

🔶 Common BiRNN: Bidirectional LSTM (BiLSTM)

- Uses **two LSTM layers**, one forward, one backward.
- Captures **context from both sides**.
- Often gives **better performance** on sequence tasks like:
  - Text classification
  - Named entity recognition (NER)
  - Machine translation
  - Speech recognition

Keras Code for BiLSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Bidirectional, LSTM, Dense
import numpy as np
```

```
model = Sequential()
model.add(Bidirectional(LSTM(64), input_shape=(10, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

- `Bidirectional(...)`: This **wraps** your LSTM layer to make it bidirectional.
- `LSTM(units=64)`: LSTM layer with 64 memory units.
- `input_shape=(10, 1)`: Each sample has **10 time steps**, with **1 feature** at each step.
  - **`input_shape = (timesteps, features)`**

🎯 Think of it like this:

Imagine your model is learning from **sensor data** collected every second.

Second (time step)	Temperature	Humidity
1	32.1	55
2	33.0	54
3	33.8	53

Here:

- **Time steps = 3** (3 rows)
- **Features = 2** (Temp and Humidity)

So, `input_shape = (3, 2)`

For Sequential Data (Text, Time Series):

```
input_shape=(TIMESTEPS, FEATURES)
```

- `TIMESTEPS`: How many words/days/measurements per sample
  (e.g., 100 words in a movie review)
- `FEATURES`: What describes each step
  (e.g., 1 for univariate time series, 300 for word embeddings)

For Images:

```
input_shape=(HEIGHT, WIDTH, CHANNELS)
```

(e.g., `(256, 256, 3)` for RGB images)

For Tabular Data:

```
input_shape=(N_FEATURES,)
```

(e.g., `(10,)` for 10 columns in your dataset)

- `Dense(1)`: Predicts a **single number**.
- `compile()`: Prepares the model to train using `adam` optimizer and MSE loss.

When to Use Bidirectional RNNs?

✅ **Best for**:

- Text classification (sentiment analysis)
- Named entity recognition
- Machine translation
- Speech recognition
- Any task where full context matters

- Time-Series Forecasting

❌ **Not ideal for**:

- Real-time predictions (can't see future in live applications)
    - Don’t use for live speech translation
- Very long sequences (computationally expensive)
- Causal prediction tasks (where future info shouldn't be seen)

Comparison: Unidirectional vs Bidirectional

Feature	Unidirectional LSTM	Bidirectional LSTM
Context	Past only	Past + Future
Accuracy	Lower	Higher
Speed	Faster	Slower (~2x)
Memory Usage	Lower	Higher
Live Prediction	Possible	Not possible