Understanding LSTMs and Bidirectional LSTMs for Text Classification

Introduction to RNNs

Think of reading a book: we understand words in context, not in isolation. Traditional neural networks fail here as they have no memory. Recurrent Neural Networks (RNNs) fix this with loops that pass information forward, but they suffer from **short-term memory** due to the *vanishing gradient problem*.

What is an LSTM?

LSTMs (Long Short-Term Memory networks) solve this by adding a **cell state** (long-term memory) and **gates** that control information flow:

- **Forget Gate** → Decides what to discard.
- **Input Gate** → Decides what new info to store.
- **Output Gate** → Decides what to use for output.

Each gate uses a sigmoid (0–1 like a water valve). The network *learns* when to open/close gates during training.

The important point here is the fact that gates learn from data when to open and close themselves using sigmoid functions.

So how do gates actually work?

For instance, if we were predicting stock prices:

If today's market news is just noise, the forget gate might output a value close to 0 for that information, effectively deleting it from the cell's memory.

If a major earnings report is released, the input gate might output a value close to 1, writing this crucial new information into memory.

Finally, the output gate controls how much of the internal memory is used to make the prediction for the current time step.

So, the "activation" of these gates is not a manual process. It’s a dynamic, data-driven system where learned weights allow the gates to react intelligently to the sequence as it comes in.

Key LSTM Parameters

- **Units (Hidden Size):** Number of neurons (memory capacity).
- **Layers:** Stack multiple LSTMs → word → sentence → paragraph.
- **Sequence Length:** How many steps the model looks back.
- **Batch Size:** How many sequences per update.
- **Learning Rate:** Speed of learning.
- **Dropout:** Randomly drops neurons → avoids overfitting.
- We can always use callbacks with the model for dynamic learning rate increase or decrease.

Example from Notebook

```
model = Sequential([
    Embedding(vocab_size, embedding_dim),
    LSTM(64, return_sequences=False),
    Dropout(0.5),
    Dense(1, activation="sigmoid")
])
```

What is a BiLSTM?

Standard LSTM → reads left to right only. But meaning often depends on *future* words too.

BiLSTM = Two LSTMs in parallel:

- Forward LSTM (left → right)
- Backward LSTM (right → left)

Outputs are combined → richer context.

Example from Notebook

```
model = Sequential([
    Embedding(vocab_size, embedding_dim),
    Bidirectional(LSTM(128, return_sequences=True)),
    Bidirectional(LSTM(64)),
    Dense(1, activation="sigmoid")
])
```

Results

Validation Accuracy Comparison

Model Accuracy Comparison

```
--- Model Accuracy Comparison ---
1. Logistic Regression:           78.00%
2. Naive Bayes:                   80.17%
3. Simple LSTM:                   54.71%
4. Advanced Bidirectional LSTM:   95.30%
```

Why BiLSTM wins:

- Uses both past & future context.
- Deeper stacked architecture.
- Larger memory capacity.
- LSTM was not deep enough to compete with BI directional LSTM in our scenario

---

Reality Check

BiLSTM is not always the best choice:

- **Best when:** Context from both directions matters (sentiment, translation).
- **Not worth it when:** Data is naturally one-directional (stock prediction).

Most impactful hyperparameter tuning:

- Sequence length
- Units (hidden size)
- Dropout (prevents overfitting)
- Learning rate (sweet spot ≈ 0.001)

Future Steps for our BFRB datasets

- Models and strategies similar to LSTMs: GRUs and Vanilla RNN. We can try these out and see how they fair on the datasets
- Using transformers and the concept of attention for more accuracy and understanding of the videographic data.
- Understanding the parameters that LSTMs and Bi LSTMs have fine tuning them for more accuracy. Target parameters include: Sequence Length, Hidden size, dropout and learning rate, even using callbacks to increase and decrease learning rates on plateaus.
- Experimenting with ensembling the transformer model with a Convolutional layer 1D model combined with Bi LSTMs to get the best of both RNN and transformer worlds.

Conclusion

- **RNN → limited memory**
- **LSTM → long-term memory with gates**
- **BiLSTM → two-way context, much better accuracy for text classification**

For our project: BiLSTM clearly dominates LSTM, but with higher cost. Choosing the right model = balancing accuracy vs compute.

reference to the notebook: https://colab.research.google.com/drive/1f1lsOOWy0dgQ-8KEEP_CTiaxcRDRi2fX?usp=sharing

Kaggle dataset used in the notebook: https://www.kaggle.com/competitions/nlp-getting-started\ \ future reading on transformers talked about in this document: https://arxiv.org/pdf/1706.03762v6