Understanding LSTMs: How Long Short-Term Memory Networks Overcome the Vanishing Gradient Problem

Introduction

Many modern machine-learning tasks, including language modelling, speech recognition, and time-series forecasting, require models that can understand how information evolves over time, something feedforward networks are simply incapable of doing. Recurrent Neural Networks incorporate recurrent connections to enable past information to influence future predictions. However, standard RNNs suffer from the vanishing gradient problem, where gradients shrink exponentially during backpropagation, causing the network to "forget the start of the sequence" (Bengio et al., 1994), as highlighted in the module slides. Long Short-Term Memory networks (Hochreiter & Schmidhuber, 1997) were designed to solve exactly this issue with a gated memory system comprising a cell state and forget, input, and output gates. The gates control the flow of information and allow stable gradient propagation for very long sequences. For this reason, LSTMs have become a cornerstone in applications such as machine translation, sentiment analysis, and sequence generation.

Background and Motivation

When dealing with sequential data, such as text, audio, and temporal signals, one wants to have models to make sense of each input in terms of what has come before. The module slides show this for sentences and temporal signals, indicating that order matters. A classic RNN updates its hidden state by

$$h_t=\tanh(W_x x_t+W_h h_{t-1}),$$

meaning that every output is computed based on the current input as well as the hidden state from the previous time step. When “unrolled” or “unfolded” through time, an RNN becomes a deep computational graph, since there are many time steps at which the computation is performed, which causes problems when backpropagating through many time steps.

Because tanh and sigmoid types of activation functions have derivatives less than one, repeated multiplication of these derivatives makes gradients approach zero, known as the vanishing gradient problem. Hochreiter 1991, and Bengio et al. 1994 mathematically showed that this prevents RNNs from learning dependencies spanning many time steps. This is why the lecture slides said that RNNs "forget the start of the sequence." Thus, standard RNNs do well on short-range patterns but fail in tasks requiring long-term context, hence motivating the study of architectures such as LSTMs that can maintain stable memory over time.

Long Short-Term Memory (LSTM) Networks

LSTMs were introduced to address the inability of standard RNNs to learn long-term dependencies. Intuitively, an LSTM adds a cell state-a dedicated memory channel-and incorporates gates that control what information is stored, forgotten, or revealed. In Colah's visual explanation, the cell state is compared to a nearly linear information highway, as long-range information can flow with minimal modification unless gates intervene. This matches the course slides on "Controlling memory - Multiple Gates," which present LSTMs as RNNs augmented with additional pathways for memory control.

A generic LSTM at every time step maintains two vectors: cell state C_t, which is the long-term memory, and the hidden state h_t, the short-term output. Three gates regulate memory:

Forget gate:

$$f_t=\sigma(W_f x_t+U_f h_{(t-1)}+b_f,$$

which means deciding what prior cell state information to get rid of.

Input gate and candidate update:

$$i_t=\sigma(W_i x_t+U_i h_{(t-1)}+b_i)$$

$$\tilde{C}_t=\tanh(W_C x_t+U_C h_{(t-1)}+b_C),$$

deciding what new information enters memory.

Output gate:

$$o_t=\sigma(W_o x_t+U_o h_{(t-1)}+b_o,$$

determines how much the cell state influences the hidden state.

This architecture corresponds precisely to the four pathways in the module’s LSTM slides.

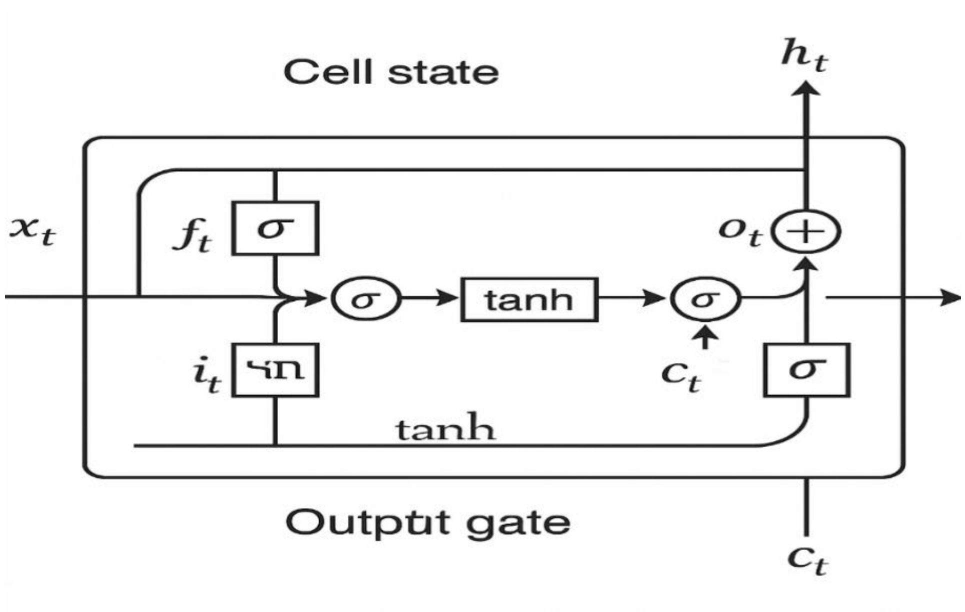
The cell state is updated additively:

$$C\_t=f\_t \odot C\_(t-1)+i\_t \odot \tilde{C}\_t,$$

which is necessary, since additive updates enable gradient flow to stay stable for very long sequences. If the values of the forget gates remain close to one, information can persist for hundreds of time steps—a situation which is impossible in ordinary RNNs. A few empirical studies show that LSTMs outperform tanh-based RNNs on tasks that require memorization over long ranges, while GRUs provide a lightweight alternative with similar benefits.

Visual Intuition: The LSTM Cell

An intuitive way to think about an LSTM is to consider the cell state as a kind of conveyor belt that runs straight through time, carrying information across time steps with minimal modifications. The three types of gates can be viewed as a kind of valve along this conveyor belt, selecting which information to discard, add, or present. Hochreiter & Schmidhuber originally called this the Constant Error Carousel, allowing gradients to just flow, without ever vanishing. This image corresponds to the module slides showing the LSTM as a memory-controlled RNN, with four parallel streams of information. Goodfellow, Bengio and Courville note especially that unlike for standard RNNs, which simply keep transforming their entire state, LSTMs distinguish between long-term storage and short-term representation, with the cell state taking the former role and the hidden state the latter, hence robustly supporting long-range memory.



.

Mathematical Framework of LSTMs

The LSTM's mathematical structure is built around gated control of information flow. At each time step:

$$\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f),\\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i),\\
\tilde{C}_t &= \tanh(W_C x_t + U_C h_{t-1} + b_C),\\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o).
\end{aligned}$$

The **cell update** is

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t,$$

and the hidden state is

$$h_t = o_t \odot \tanh(C_t).$$

This additive structure preserves gradients:

$$\frac{\partial C_t}{\partial C_{t-k}} = \prod_{j=t-k+1}^{t} f_j\,,$$

so if forget-gate values are near 1, gradients remain strong even over long sequences. This directly addresses the issue shown in the "Gradient Troubles" slide. As a result, LSTMs are significantly more effective than RNNs at modelling long-term dependencies.

Why RNNs Forget the Beginning of the Sequence

While in a standard RNN, the hidden state is updated using

h_t=tanh(W_x x_t+W_h h_(t-1)),

and in the backward pass, gradients earlier in time require repeatedly multiplying derivatives less than one. Hochreiter (1991) and Bengio et al. (1994) proved that this leads to exponential vanishing of gradients, effectively blocking the ability of the network to learn from distant past inputs. In practice this means that RNNs place disproportionate weight on recent inputs while "forgetting" early ones even though the recurrence structure should allow for information flow across many steps. This is exactly what the module's "Vanishing Gradients" slide demonstrates. Although the reverse problem of exploding gradients can also be an issue, it is easily mitigated with gradient clipping, but as discussed here vanishing gradients fundamentally limit RNN performance on long sequences, making LSTMs the primary alternative.

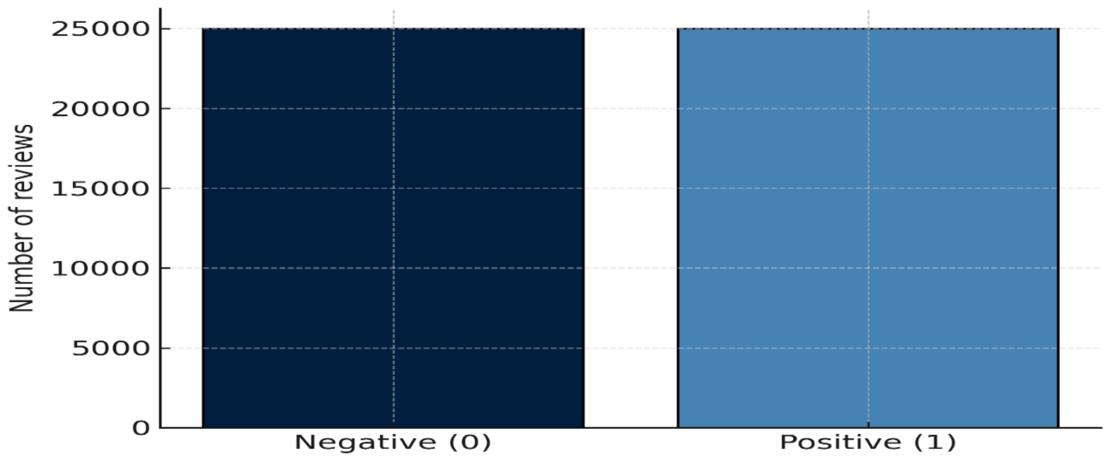
LSTM vs RNN

The table below highlights how LSTMs improve upon standard RNNs by solving the vanishing-gradient issue and enabling long-term memory through gated control mechanisms.

Feature	**RNN**	**LSTM**
Gate Structure	None	Forget, Input, Output + Candidate Update
Memory Components	Hidden state only	Hidden + cell state
Long-Term Dependency Handling	Poor	Excellent
Gradient Flow	Unstable; vanishing gradients	Stable via additive updates
Complexity	Simple	More complex
Parameter Count	Low	High
Training Stability	Poor on long sequences	Stable
Training Speed	Faster	Slower
Typical Use Cases	Short patterns	NLP, speech, time-series
Weaknesses	Forgets early info	Higher compute cost

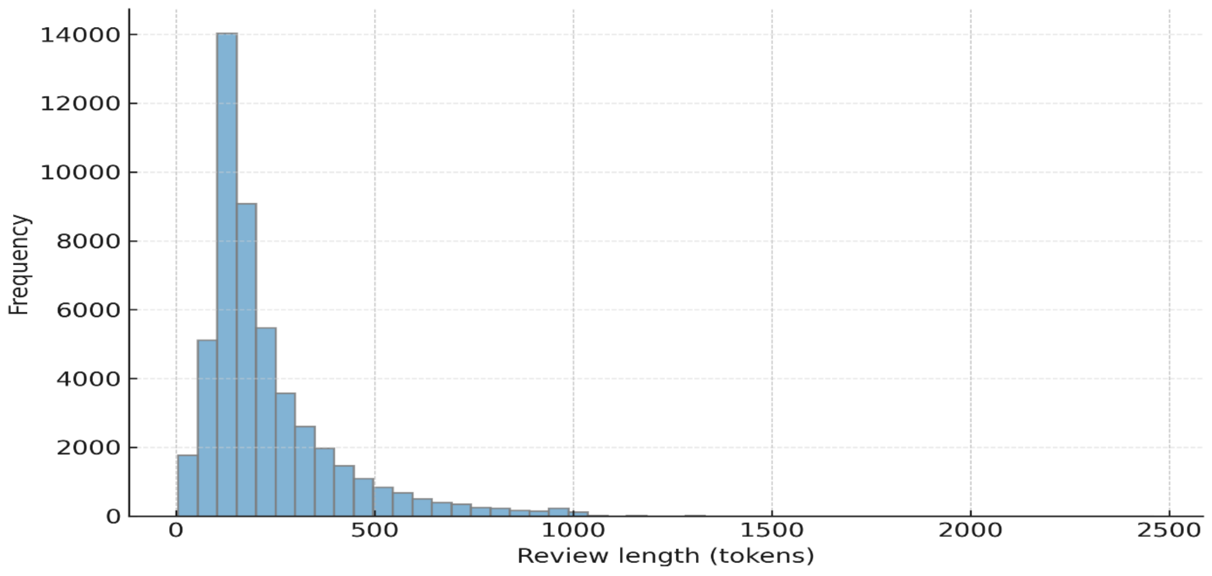
Visualisations and Figures

Sentiment Distribution Plot



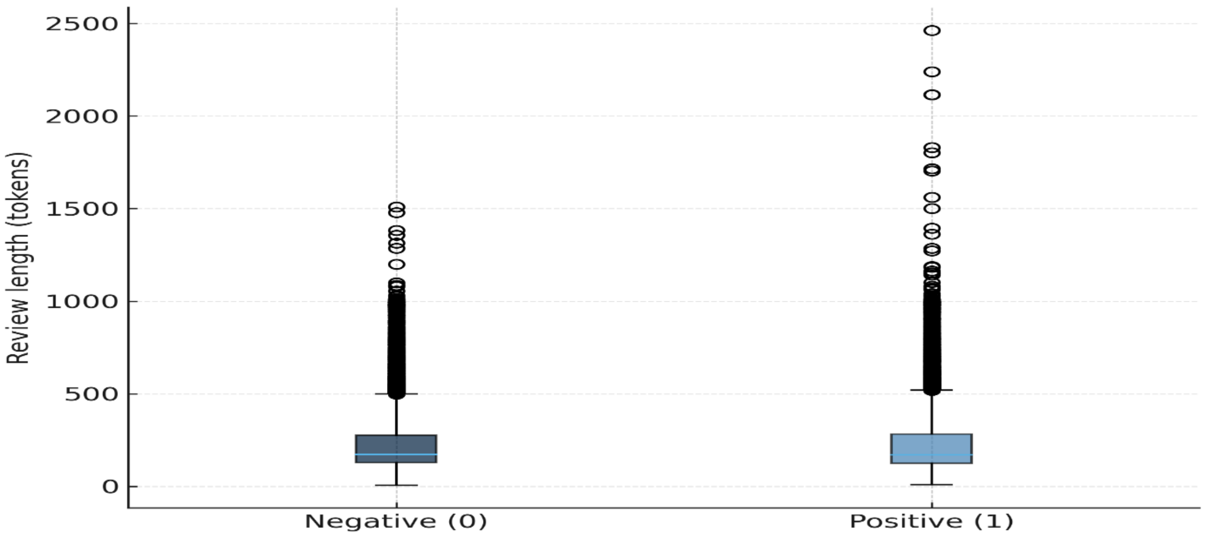
The dataset is perfectly balanced, containing an equal number of positive and negative movie reviews. This ensures that model training is not biased toward either sentiment class.

Histogram of Review Lengths



Most IMDB reviews fall between 100 and 300 tokens, with a long tail of much longer reviews extending beyond 1,000 tokens. This skewed distribution indicates why padding and fixed sequence lengths are necessary for LSTM models.

Boxplot of Review Length by Sentiment



Both positive and negative reviews have similar distributions in length, even though positive reviews are slightly more variable. The presence of extreme outliers in both classes underlines the requirement for the sequence truncation during LSTM preprocessing.

Applications of LSTMs

• Sentiment Analysis: It keeps track of full sentence meaning by preserving early contextual information.

• Next-Word Prediction / Language Modelling: This captures long-range grammatical and semantic dependencies.

• Speech and Audio Processing: Models temporal structures of sequential audio data.

• Time-Series Forecasting: Learns long-term seasonal or trend-based patterns.

• Machine Translation: Maintains long-distance linguistic relationships across clauses.

• Sequence Classification: Effective when entire sequences determine class labels.

Conclusion

Long Short-Term Memory (LSTM) networks are a powerful, theoretically justified solution to the problem of vanishing gradients preventing RNNs from learning long-term dependencies. Their gated architecture and the stable pathway along the cell state allow them to retain meaningful information across very long sequences, making them one of the most effective tools for sentiment analysis, language modelling, and time-series prediction. Further supported by the need to analyse IMDB datasets, reviews have widely varying lengths, with a reliance on context spread over many tokens and understanding, which is hard for simple RNNs to capture. By keeping gradient flow stable and meanwhile conducting selective memory management, LSTMs never fail to outperform basic recurrent models and remain a core tool for sequential data processing, even beyond the emergence of newer architectures like GRUs and Transformers. Knowing mathematically and intuitively how LSTMs work equips the practitioner with a robust and versatile tool for building accurate, context-aware machine-learning systems.

References

Bengio, Y., Simard, P. & Frasconi, P., 1994. *Learning long-term dependencies with gradient descent is difficult*. IEEE Transactions on Neural Networks, 5(2), pp.157–166.

Cho, K. et al., 2014. *Learning phrase representations using RNN encoder–decoder for statistical machine translation*. EMNLP, pp.1724–1734.

Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Learning*. MIT Press.

Hochreiter, S., 1991. *Untersuchungen zu dynamischen neuronalen Netzen*. TUM.

Hochreiter, S. & Schmidhuber, J., 1997. *Long short-term memory*. Neural Computation, 9(8), pp.1735–1780.

Olah, C., 2015. *Understanding LSTM Networks*.

Xiao, H., Rasul, K. & Vollgraf, R., 2017. *Fashion-MNIST*. arXiv:1708.07747.