

LSTM Networks: Teaching Machines Long-Term Memory

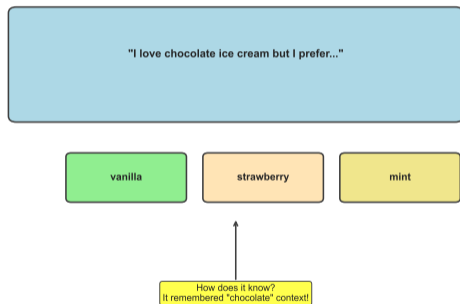
A 10-Page BSc Introduction - Zero Prerequisites

NLP Course 2025

September 27, 2025

The Problem: Predicting What Comes Next

Your Phone Predicts the Next Word



Technical Terms Explained:

What You Type:

"I grew up in Paris. I went to school there for 12 years. I

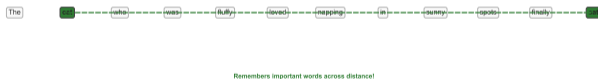
Sequence Modeling: Predicting the next element based on all previous

N-gram Models: The Baseline (And Why They Don't Work)

N-Gram: Fixed 2-Word Window (Forgets "cat"!) !



LSTM: Selective Memory (Remembers "cat"!) !



What N-grams Do:

- Count word sequences in training data
- Look at last 1-2 words only
- Pick most common next word

Notation Explained:

N-gram: A sequence of N words. Bigram = 2 words ("I love"), Trigram = 3 words ("I love chocolate")

$P(w_t \mid w_{t-1})$: Probabil-

Insight from Human Reading

Human Memory Example:

Chapter 1: "Alice was born in London in 1985. She had a happy childhood."

Chapter 3: "After graduating from university, Alice moved to New York."

Chapter 7: "Now 38 years old, Alice reflected on her life in ---"

What You Remember:

- Alice (main character) [YES]
- Born in London [YES]
- Moved to New York [YES]
- Currently 38 [YES]

What You Forgot:

- "had a happy childhood" [NO]
- "graduating from university" [NO]
- Exact wording [NO]

Three Mechanisms We Need:

1. Forget Gate: Decide what to remove from memory
Example: Forget "chocolate" after period

2. Input Gate: Decide what to store
Example: Store "Paris" strongly

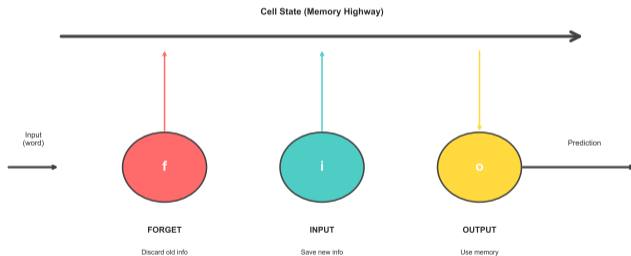
3. Output Gate: Decide what to use now
Example: Recall "Paris" when predicting language

Technical Terms:

Gate: A learned decision mechanism that outputs values between 0 (block) and 1 (allow). Acts like a controllable valve

Long Short-Term Memory: Gated Memory Cells

LSTM Cell: Three Gates Control Memory

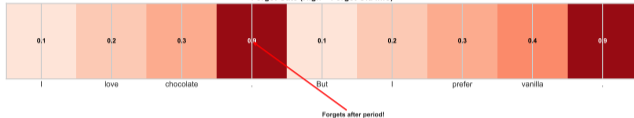


Like Traffic Lights: Red (forget) • Green (input) • Yellow (output)

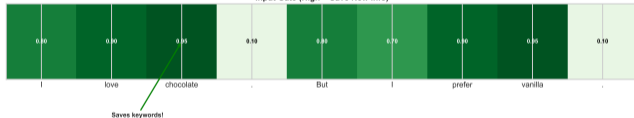
Gate Mechanisms with Concrete Examples

LSTM Gate Activations: "I love chocolate. But I prefer vanilla."

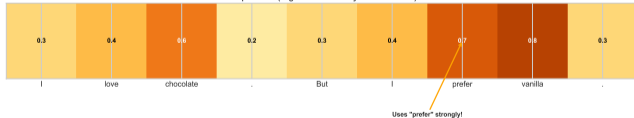
Forget Gate (High = Forget Old Info)



Input Gate (High = Save New Info)



Output Gate (High = Use Memory for Prediction)



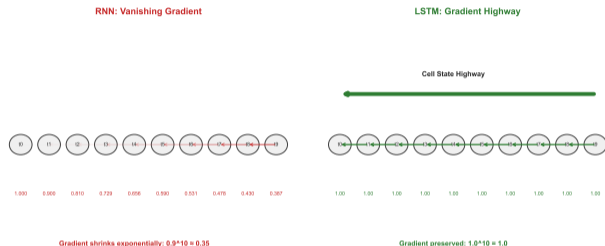
Forget Gate f_t :

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

What This Chart Shows:

- Real gate values over sentence

Why LSTMs Can Remember 50-100+ Steps



The Update Equation:

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

In plain English: New memory = (keep some old) + (add some new). The \odot means multiply each number separately

Why Addition is Magic:

- Old C_{t-1} directly adds to new C_t

Comparison - RNN vs LSTM:

RNN (Multiplicative):

$$h_t = \tanh(W_h h_{t-1} + \dots)$$

After 50 steps:

- Signal: $0.5^{50} \approx 10^{-15}$
- Information lost!

LSTM (Additive):

The Full Forward Pass (One Time Step)

All Six Equations:

$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$	Forget gate
$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$	Input gate
$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$	Candidate
$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$	Cell update
$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$	Output gate
$h_t = o_t \odot \tanh(C_t)$	Hidden state

Activation Functions:

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$

- Range: (0, 1)
- Used for gates (0 = close, 1 = open)

Tanh: $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Concrete Numerical Example:

Input: word "love" (after "I")

Step 1: Compute gates

- $f_t = [0.62, 0.45, 0.69, \dots]$ (keep some)
- $i_t = [0.77, 0.69, 0.38, \dots]$ (add much)
- $o_t = [0.71, 0.75, 0.45, \dots]$ (reveal most)

Step 2: Create candidate

- $\tilde{C}_t = [0.54, -0.29, 0.72, \dots]$

Step 3: Update cell state

- Old: $C_{t-1} = [0.5, 0.3, 0.2, \dots]$
- Keep: $f_t \odot C_{t-1} = [0.31, 0.14, 0.14, \dots]$
- Add: $i_t \odot \tilde{C}_t = [0.42, -0.20, 0.27, \dots]$
- New: $C_t = [0.73, -0.06, 0.41, \dots]$

Step 4: Compute output

Backpropagation Through Time (BPTT)

LSTM Training: Watching It Learn

Epoch 1: Random Initialization

Input: "I love chocolate"

Prediction: "xjwkq"

Loss: 8.5 (Gibberish)

Epoch 10: Learning Letters

Input: "I love chocolate"

Prediction: "cream"

Loss: 2.1 (Better!)

Epoch 50: Understanding Context

Input: "I love chocolate"

Prediction: "ice cream"

Loss: 0.4 (Good!)

Epoch 200: Fluent Generation

Input: "I love chocolate"

Prediction: "Ice cream
and strawberry cake"

Loss: 0.08 (Excellent!)

Technical Terms:

BPTT: Backpropagation
Through Time. Compute

What This Shows:

Applications:

NLP: Translation, generation, sentiment

Speech: Recognition (Siri), music generation

Time Series: Stock, weather, energy, healthcare

Comparison: N-gram vs RNN vs LSTM

Feature	N-gram	RNN	LSTM
Memory Type	Fixed window	Fading	Selective
Long Context	No	Partial	Yes
Parameters	Few	Moderate	Many
Training Speed	Fast	Medium	Slow
Handling Gradient	No	Yes	Solved
Best For	Short (2-3 words)	Medium (10 words)	Long (50+ words)
Example	"I love."	"The cat sat."	"The cat, who was, finally."

Key Insight: LSTMs still relevant in 2025! Complementary to Transformers.

Where LSTMs Are Used + Summary

Equations Reference:

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

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

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

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

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

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

Summary:

- **Problem:** 50-100 step memory needed
- **Solution:** 3 gates + additive cell state
- **Impact:** Google Translate, foundation for Transformers

When to Use:

LSTMs:

- Time series (SOTA)
- Real-time
- Mobile/edge
- Limited data

Transformers:

- Large datasets
- Parallel training
- Bidirectional
- SOTA NLP

Notation:

- t : time, x_t : input
- h_t : hidden, C_t : cell
- σ : sigmoid (0-1)