

# Sentiment Analysis with Transformers

## From Words to Understanding

BSc NLP Module

November 2025

**After this lecture, you will be able to:**

1. Explain why bidirectional context improves sentiment analysis
2. Describe the BERT fine-tuning process for classification tasks
3. Interpret performance metrics and attention patterns
4. Evaluate tradeoffs between traditional ML and transformer approaches
5. Identify when BERT-based sentiment analysis is appropriate

## The Puzzle: When “Good Accuracy” Isn’t Good Enough

Your startup processes movie reviews daily. You build a BOW+SVM classifier with good accuracy. Investors are thrilled!

**But then...** users start complaining:

*“Great, another boring superhero movie”* → Model: **POSITIVE** → **User complaint!**

---

**Mystery Question:** If accuracy is “good,” why are users complaining? Let’s investigate...

## Clue #1: Why Traditional Methods Fail

Review Text	BOW Predicts	Actual	Why BOW Fails
"Great, another boring movie"	POSITIVE	NEGATIVE	Sees "Great", ignores context
"This is not a bad film"	NEGATIVE	POSITIVE	Sees "bad", ignores "not"
"Absolutely incredible" vs "Somewhat good"	Both POSITIVE	Strong vs Weak	Cannot measure intensity

### Pattern Emerges:

- **Sarcasm:** Word polarity reversed by context
- **Negation:** Modifier words change meaning
- **Intensity:** Strength requires understanding relationships

BOW treats words as independent. Context, word order, and relationships matter!

## Traditional: BOW + SVM

### Pipeline:

1. Input: "Great, another boring movie"
2. Word Counts: Great=1, another=1, boring=1, movie=1
3. Feature Vector: [1, 1, 1, 1, ...]
4. SVM Classifier
5. Output: **POSITIVE** (WRONG!)

**Problem: No word order, no context!**

## Breakthrough: BERT

### Pipeline:

1. Input: "Great, another boring movie"
  2. Tokenization + [CLS] token
  3. Transformer Layers (Bidirectional)
- CLS** = Sentence Representation
4. Output: **NEGATIVE** (CORRECT!)

**Solution: Understands context!**

---

Structural limitation: BOW/TF-IDF can't capture word order or bidirectional context. BERT can.

# Breakthrough: BERT for Sentiment Analysis

**Key Insight:** Bidirectional context solves our puzzle!

## BERT Architecture (High-Level)

- Input: Tokenize with [CLS] token
- Transformer layers (12-24)
- Bidirectional attention

**CLS** = sentence representation

- Classification head on [CLS]

## How It Solves Our Puzzle

- Sarcasm: “Great” seen with “boring”
- Negation: “not” modifies “bad”
- Intensity: Measures strength context
- Word order: Fully preserved
- Relationships: Captured by attention

---

**Breakthrough:** BERT processes words in both directions, understanding context like humans do.

# The Solution: BERT Fine-Tuning Pipeline

## Four-Stage Process:

1. **Stage 1: Pre-trained BERT** (Already done for us!)
  - Trained on Wikipedia + Books (general language)
  - Knows grammar, context, meaning relationships
2. **Stage 2: Add Classifier Head**
  - Linear layer with random initialization
  - 2 outputs: Positive vs Negative
3. **Stage 3: Fine-Tune on Sentiment Data**
  - Train on IMDb reviews (labeled data)
  - Needs only 1000s of examples (not millions!)
4. **Stage 4: Deploy**
  - Apply to new reviews in production
  - Real-time predictions

---

**Key insight:** Pre-training gives general understanding, fine-tuning adapts to sentiment task. We don't start from zero!

### Quick Check: Match the stages to what's learned

1. Pre-trained BERT learns: \_\_\_\_\_
2. Adding classification head: \_\_\_\_\_
3. Fine-tuning on IMDb: \_\_\_\_\_
4. Deployment: \_\_\_\_\_

### Options:

- A. General language understanding (context, grammar, meaning)
- B. Random initialization for sentiment classification
- C. Sentiment-specific patterns from labeled data
- D. Apply to new reviews in production

---

Answers: 1-A, 2-B, 3-C, 4-D. Understanding stages prevents “train from scratch” misconception.

## Loss Function (Intuition)

- Cross-entropy loss
- Pushes probability toward correct label
- POSITIVE: maximize  $P(\text{pos})$
- NEGATIVE: maximize  $P(\text{neg})$
- Gradient descent updates BERT weights

## Resource Requirements

- **Data:** Thousands of labeled examples (not millions!)
- **Compute:** Hours on GPU (not weeks)
- **Memory:** Standard GPU RAM
- **Cost:** Affordable on cloud (accessible!)

## Key Insight

Fine-tuning is efficient because:

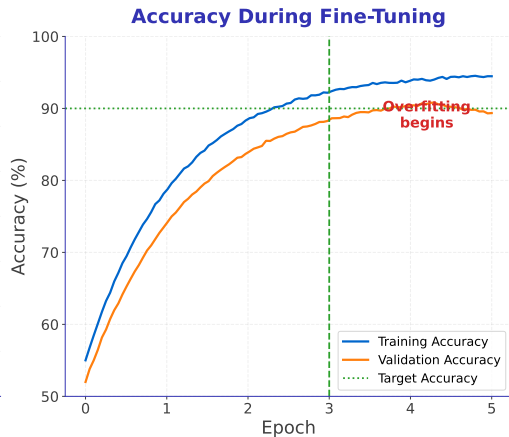
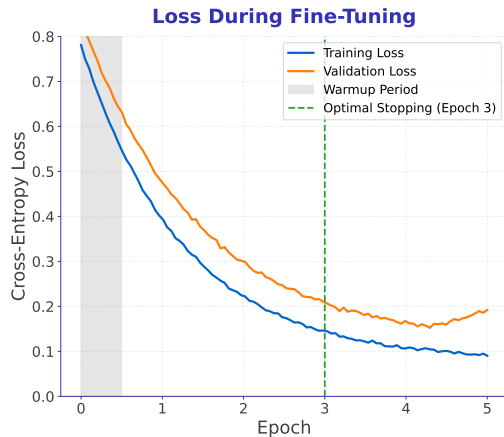
- BERT already knows language (pre-training)
- We only teach sentiment patterns (fine-tuning)
- Much faster than training from scratch

---

Practical reality: BERT fine-tuning is accessible for real projects, not just research labs.



# Training Dynamics: Loss and Accuracy Curves



Empirical validation: 3-5 epochs sufficient. Warmup prevents destroying pre-trained weights. Overfitting signals when to stop.

## Performance Comparison on Sentiment Benchmarks:

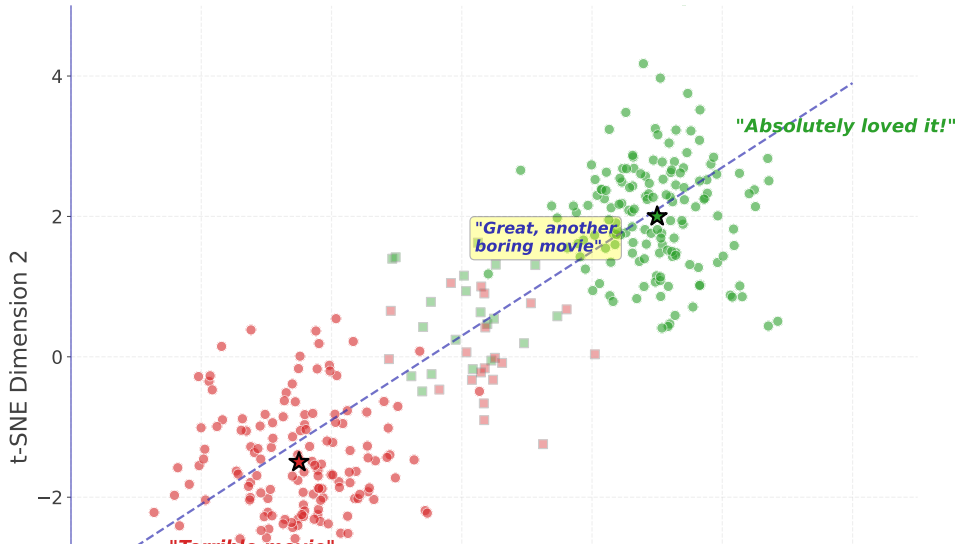
Method	Relative Performance	Category
BOW + SVM	Baseline	Traditional
TF-IDF + Logistic	Slightly better	Traditional
LSTM	Moderate improvement	Neural
BERT-base BERT-large	<b>Large improvement</b> <b>Best performance</b>	Transformer Transformer

## Key Observations:

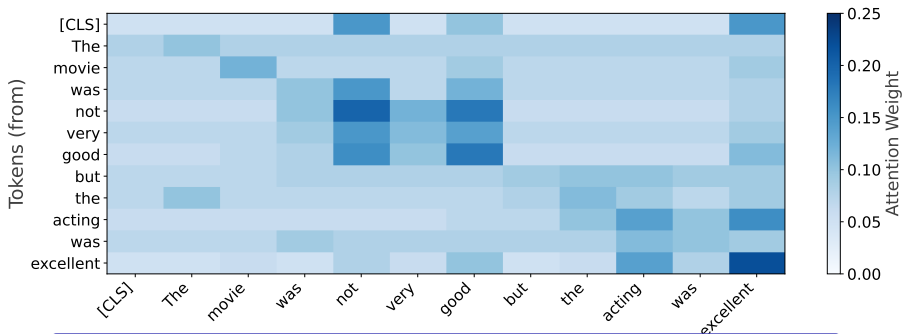
- Traditional methods: Limited by bag-of-words assumption
- BERT: Substantial improvement over all baselines
- Bidirectional context (our breakthrough) accounts for gains

Empirical validation: BERT significantly outperforms traditional methods on sentiment benchmarks. Mystery solved!

## BERT [CLS] Embedding Space: Sentiment Clusters



## Understanding the Magic: BERT Attention Heatmap



**BERT focuses on "not", "good", "excellent" - understands negation and contrast**

BERT's attention reveals how it solves our puzzle: focusing on critical context words like "not", "good", "excellent".

# Wisdom: When to Use BERT vs Traditional Methods

Use BERT When...	Use Traditional (BOW/TF-IDF) When...
Context is critical (sarcasm, negation) You have 1000+ labeled examples Accuracy matters more than speed GPU resources available Complex language understanding needed	Simple patterns suffice (keyword matching) Limited labeled data (<100 examples) Need real-time inference (<1ms) CPU-only deployment Interpretability critical

## Key Tradeoffs:

- **Speed:** BERT slower per review vs BOW (near-instant)
- **Data:** BERT needs more labeled examples than traditional
- **Accuracy:** BERT substantially better F1-score
- **Interpretability:** Traditional clearer, BERT needs attention analysis

Engineering wisdom: Choose method based on constraints (data, compute, latency) not just “best” performance.

### Binary Cross-Entropy for Sentiment:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Where:

- $y_i \in \{0, 1\}$ : True label (0=negative, 1=positive)
- $\hat{y}_i \in [0, 1]$ : Predicted probability
- $N$ : Number of training examples

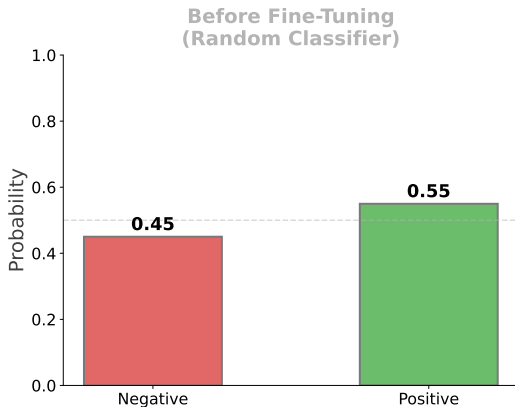
### Intuition:

- When  $y_i = 1$  (positive): Loss =  $-\log(\hat{y}_i)$ 
  - If  $\hat{y}_i = 0.9$  (confident): Loss  $\approx 0.05$  (small)
  - If  $\hat{y}_i = 0.1$  (wrong): Loss  $\approx 2.3$  (large)
- Gradient descent minimizes loss by adjusting BERT weights
- Optimizer: AdamW with small learning rate (typical for fine-tuning)

---

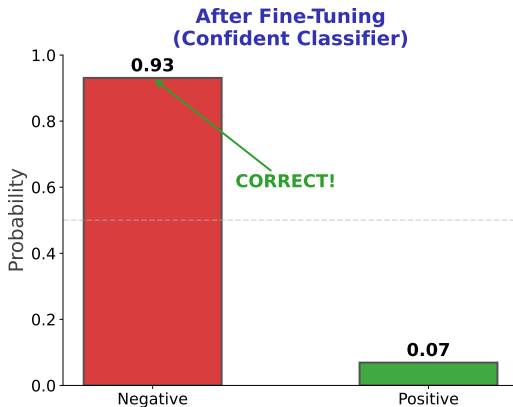
Cross-entropy penalizes confident wrong predictions more than unconfident wrong ones.

## From Logits to Predictions: Softmax & Cross-Entropy



**Softmax Calculation (Worked Example)**

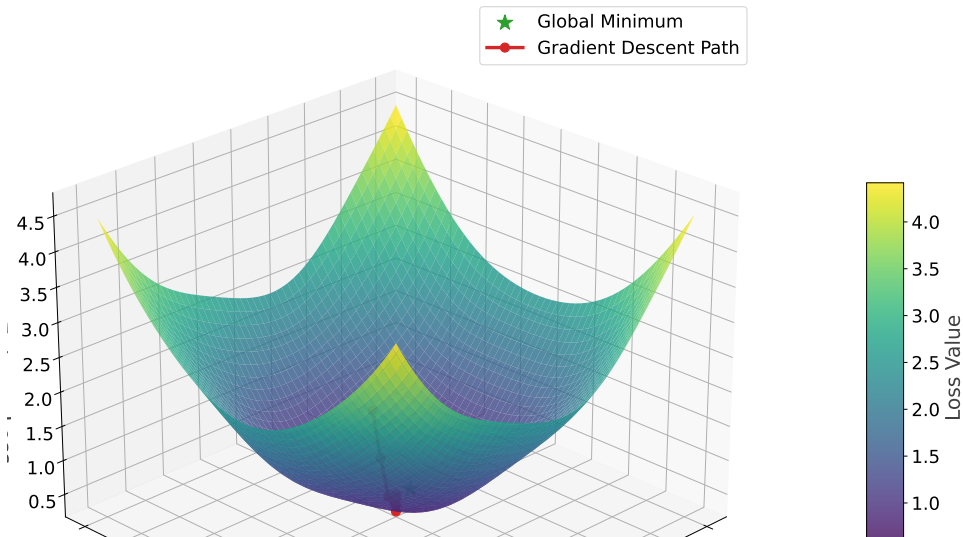
Input: "Great, another boring movie"



**Cross-Entropy Loss (Worked Example)**

Ground Truth:  $y = [1, 0]$  (Negative)  
Predicted:  $\hat{y} = [0.93, 0.07]$

## Loss Landscape: Fine-Tuning BERT for Sentiment





### AdamW Optimizer:

- Adam with weight decay
- Adaptive learning rates per parameter
- Standard hyperparameters work well for most tasks
- Weight decay prevents overfitting

### Learning Rate Schedule:

- **Warmup:** Linear increase from 0 to peak (first portion of training)
- **Decay:** Linear decrease to 0 over remaining steps
- **Peak LR:** Small learning rate (typical for fine-tuning)
- **Why:** Prevents catastrophic forgetting of pre-trained weights

### Batch Sizes & Training:

- Batch size: Limited by GPU memory
- Epochs: Few epochs sufficient (more can cause overfitting)
- Gradient accumulation if GPU memory limited

---

Careful hyperparameter tuning prevents destroying pre-trained knowledge while learning sentiment.

## Appendix A2b: Attention Score Calculation

**Example:** “The movie was not very good”

**Step 1: Create Query, Key, Value vectors**

- Query (“not”):  $\mathbf{q} = [0.8, 0.2, -0.1]$
- Key (“good”):  $\mathbf{k} = [0.7, 0.3, 0.1]$

**Step 2: Compute attention score**

$$\text{score}(\text{not}, \text{good}) = \frac{\mathbf{q} \cdot \mathbf{k}}{\sqrt{d_k}} = \frac{0.8 \times 0.7 + 0.2 \times 0.3 + (-0.1) \times 0.1}{\sqrt{3}} = \frac{0.61}{1.73} = 0.35$$

**Step 3: Apply softmax (over all keys)**

$$\alpha_{\text{not} \rightarrow \text{good}} = \frac{e^{0.35}}{e^{0.35} + e^{0.1} + e^{0.2} + \dots} = 0.18$$

**Result:** “not” attends to “good” with weight 0.18 — this connection helps BERT understand negation!

---

Attention scores reveal HOW BERT learns to connect “not” with the word it negates.

## Appendix A3: Aspect-Based Sentiment Analysis

**Problem:** Extract sentiment per product aspect

*"The phone camera is excellent but battery life is terrible."*

- Camera: **POSITIVE**
- Battery: **NEGATIVE**

### BERT Approach:

1. Identify aspects (camera, battery) via NER or keywords
2. For each aspect:
  - Create input: [CLS] aspect [SEP] sentence [SEP]
  - Fine-tune BERT to predict sentiment for that aspect
  - Use attention to find aspect-relevant words
3. Aggregate aspect sentiments

### Applications:

- Product reviews (Amazon, Yelp)
- Survey analysis
- Brand monitoring

---

Aspect-based extends BERT from document-level to fine-grained sentiment extraction.

## Appendix A4: Multi-Label Sentiment Analysis

**Problem:** Detect multiple emotions per text

*"I'm excited about the trip but worried about the cost."*

- Joy: **HIGH**
- Anxiety: **MEDIUM**
- Anger: **LOW**

**BERT Modifications:**

- **Architecture:** Replace softmax with sigmoid activation
- **Output:**  $K$  independent probabilities (one per emotion)
- **Loss:** Binary cross-entropy for each label

$$\mathcal{L} = -\frac{1}{K} \sum_{k=1}^K [y_k \log(\hat{y}_k) + (1 - y_k) \log(1 - \hat{y}_k)]$$

- **Threshold:** Predict label if  $\hat{y}_k > 0.5$

**Applications:**

- Emotion detection (Plutchik's wheel)
- Mental health monitoring
- Customer service routing

---

Multi-label captures emotional complexity beyond simple positive/negative classification.

## Appendix A5: Zero-Shot Sentiment with Prompting

**Problem:** Classify sentiment with NO labeled data

**Approach:** Use instruction-tuned LLMs (GPT-4, Claude, Llama)

### Prompt Template

Classify the sentiment of the following review as POSITIVE or NEGATIVE.

Review: ‘‘Great, another boring superhero movie’’

Sentiment:

### Why It Works:

- Pre-trained on massive text (understands sentiment)
- Instruction-tuned to follow prompts
- Can reason about sarcasm, negation, intensity
- No fine-tuning required

### Tradeoffs vs BERT Fine-Tuning:

- **Pros:** No labeled data, handles rare cases, flexible
- **Cons:** Slower inference, higher per-query cost, less controllable

Zero-shot useful for prototyping or low-resource scenarios, but fine-tuned BERT better for production.

### Foundational Papers:

- Devlin et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers. NAACL.
- Liu et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv.
- Sanh et al. (2019). DistilBERT: A distilled version of BERT (smaller, faster).

### Practical Tools:

- Hugging Face Transformers: [transformers.huggingface.co](https://transformers.huggingface.co)
- Datasets: IMDb, SST-2, Yelp, Twitter Sentiment
- Pre-trained models: BERT, RoBERTa, DistilBERT, ELECTRA

### Tutorials:

- Hugging Face Course: [huggingface.co/course](https://huggingface.co/course)
- Google Colab notebooks (free GPU access)
- Fast.ai NLP course

### Advanced Topics:

- Adversarial robustness in sentiment analysis
- Cross-lingual sentiment (multilingual BERT)
- Temporal aspects (sentiment over time)

---

Start with Hugging Face tutorials for hands-on experience fine-tuning BERT for sentiment analysis.

### Cross-Entropy Gradient (For Backpropagation):

$$\frac{\partial \mathcal{L}}{\partial z_j} = \hat{y}_j - y_j$$

where  $z_j$  is the logit for class  $j$ . This simple gradient is why cross-entropy works so well!

### Softmax Jacobian:

$$\frac{\partial \hat{y}_i}{\partial z_j} = \hat{y}_i(\delta_{ij} - \hat{y}_j)$$

where  $\delta_{ij}$  is the Kronecker delta (1 if  $i = j$ , else 0).

### Attention Weight Gradient:

$$\frac{\partial \alpha_{ij}}{\partial \mathbf{q}_i} = \alpha_{ij} \left( \mathbf{k}_j - \sum_k \alpha_{ik} \mathbf{k}_k \right) / \sqrt{d_k}$$

### BERT Fine-Tuning Update:

$$\theta_{t+1} = \theta_t - \eta \cdot \text{AdamW}(\nabla_{\theta} \mathcal{L})$$

with small learning rate  $\eta$  and weight decay  $\lambda$  (standard fine-tuning values).

---

These gradients enable end-to-end training: error signal flows from loss through attention to word embeddings.