Sentiment Analysis with Transformers

From Words to Understanding

BSc NLP Module

November 2025

Learning Goals & The Puzzle

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 85% Isn't Good Enough

Your startup gets 10,000 movie reviews per day. You build a BOW+SVM classifier with 85% accuracy. Investors are thrilled!

But then... users start complaining:

"Great, another boring superhero movie" → Model: POSITIVE → User complaint!

Mystery Question: If 85% is good, why are users complaining? Let's investigate...

Clue #1: Why Traditional Methods Fail

Sarcasm

"Great, another boring\nsuperhero movie"

Actual: NEGATIVE

BOW Predicts:
POSITIVE

Why BOW Fails:
BOW sees "Great"\nignores context

Negation

"This is not a\nbad film"

Actual: POSITIVE

BOW Predicts:
NEGATIVE

Why BOW Fails:
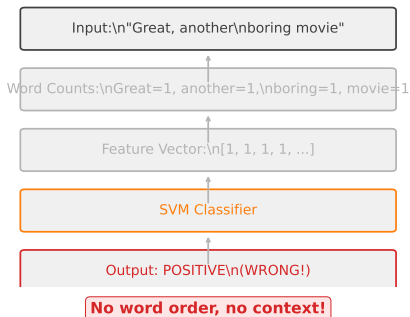BOW sees "bad"\nignores "not"

Intensity

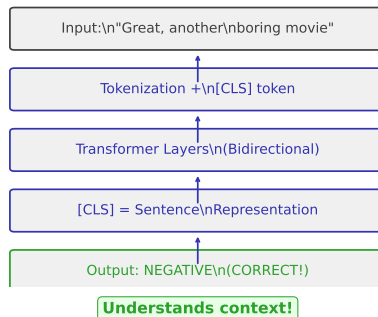*"Absolutely incredible"\nvs\n"Somewhat good"*

BOW Predicts:
BOTH POSITIVE

Dead End: Traditional Methods Can't Solve Our Puzzle

Traditional: BOW + SVM

Input:\n"Great, another\nboring movie"

↑

Word Counts:\nGreat=1, another=1,\nboring=1, movie=1

↑

Feature Vector:\n[1, 1, 1, 1, ...]

↑

SVM Classifier

↑

Output: POSITIVE\n(WRONG!)

No word order, no context!

Breakthrough: BERT

Input:\n"Great, another\nboring movie"

↑

Tokenization +\n[CLS] token

↑

Transformer Layers\n(Bidirectional)

↑

[CLS] = Sentence\nRepresentation

↑

Output: NEGATIVE\n(CORRECT!)

Understands context!

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

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
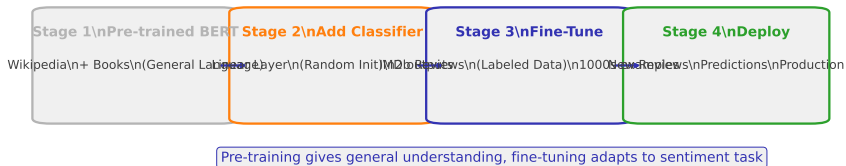
- Classification head on [CLS]

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

Stage 1\nPre-trained BERT	Stage 2\nAdd Classifier	Stage 3\nFine-Tune	Stage 4\nDeploy
Wikipedia\n+ Books\n(General Language)	Linear Layer\n(Random Init)	IMDb Reviews\n(Labeled Data)\n1000 samples	New Reviews\nPredictions\nProduction

Pre-training gives general understanding, fine-tuning adapts to sentiment task

Pre-training (general language) + Fine-tuning (sentiment task) = Efficient solution to our puzzle.

Checkpoint Quiz

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.

Training Details: Making It Practical

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
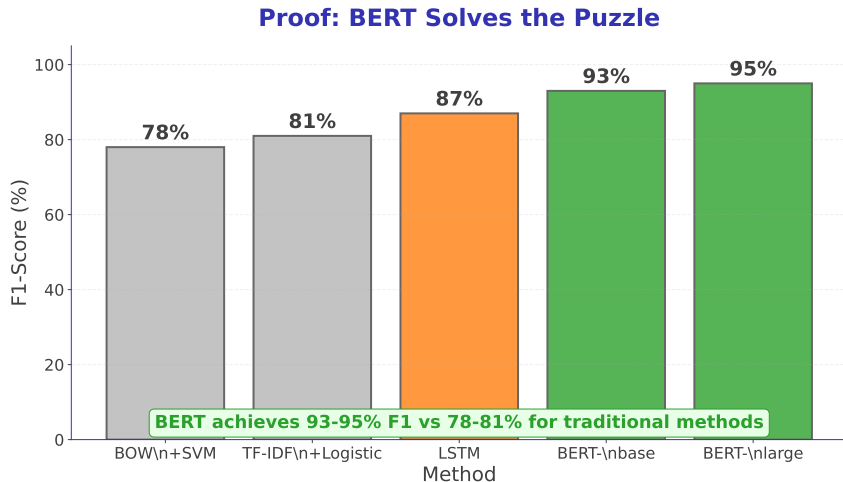
Resource Requirements
- **Data**: 1000s of labeled examples (not millions!)
- **Compute**: Hours on GPU (not weeks)
- **Memory**: 8-16GB GPU RAM
- **Cost**: $10-50 on cloud (accessible!)
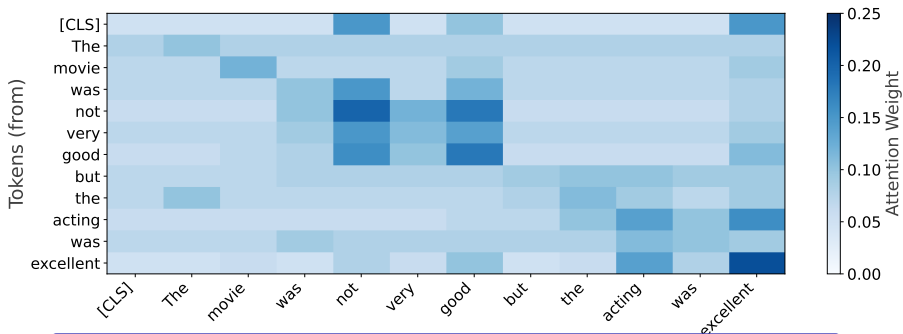
Key Insight

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.

Proof: BERT Solves the Puzzle

Empirical validation: BERT achieves 93-95% F1 vs 78-81% for traditional methods. Mystery solved!

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".

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

Key Tradeoffs:

- **Speed**: BERT $\sim$50ms/review vs BOW $\sim$1ms/review
- **Data**: BERT needs 1000s, traditional works with 100s
- **Accuracy**: BERT +10-15% F1-score improvement
- **Interpretability**: Traditional clearer, BERT needs attention analysis

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

Appendix A1: Cross-Entropy Loss (Mathematical Details)

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 ≈ 0.05 (small)
  - If $\hat{y}_i = 0.1$ (wrong): Loss ≈ 2.3 (large)
- Gradient descent minimizes loss by adjusting BERT weights
- Optimizer: AdamW with learning rate $\approx 2 \times 10^{-5}$

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

Appendix A2: Optimization & Hyperparameters

AdamW Optimizer:

- Adam with weight decay
- Adaptive learning rates per parameter
- Typical settings: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
- Weight decay: $\lambda = 0.01$

Learning Rate Schedule:

- **Warmup**: Linear increase from 0 to peak over first 10% of steps
- **Decay**: Linear decrease to 0 over remaining steps
- **Peak LR**: 2×10^{-5} to 5×10^{-5}
- **Why**: Prevents catastrophic forgetting of pre-trained weights

Batch Sizes & Training:

- Batch size: 16-32 (limited by GPU memory)
- Epochs: 3-5 (more can cause overfitting)
- Gradient accumulation if GPU memory limited

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

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 (100-1000ms), expensive ($0.001-0.01/review), less controllable

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

Appendix A6: Resources & Further Reading

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`
- Datasets: IMDb, SST-2, Yelp, Twitter Sentiment
- Pre-trained models: BERT, RoBERTa, DistilBERT, ELECTRA

Tutorials:
- Hugging Face Course: `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.