Automated Customer Review Sentiment Analysis

NLP Project - Traditional ML vs Transformers

Table of Contents

1. Introduction & Business Case

Problem Statement

- Retail companies receive **thousands of text reviews monthly**
- Manual categorization is time-consuming and costly
- Need for **automated sentiment classification**

Key Challenges

- Users don't always leave numeric scores
- Different users interpret ratings differently (4/5 varies by user)
- Need real-time insights into customer sentiment

Project Goals

✓ Classify reviews as **Positive, Negative, or Neutral**
✓ Compare **Traditional ML** vs **Deep Learning (Transformers)**
✓ Evaluate which approach yields better results

2. Data Collection & Preparation

Dataset Overview

- **Total reviews:** 55,000 (IMDB + Amazon local datasets)
- **After cleaning:** 54,615 reviews
- **Sources:** HuggingFace IMDB + Local Amazon CSVs

Sentiment Mapping Logic

- **Ratings 1, 2, 3** → Negative
- **Rating 4** → Neutral
- **Rating 5** → Positive

Sentiment Distribution

Sentiment	Count	Percentage
Positive	33,252	60.5%
Negative	14,958	27.2%
Neutral	6,776	12.3%

Handling Class Imbalance

- Class weights applied (balanced/balanced_subsample)
- Mild oversampling for Neutral class
- Test set kept untouched for honest evaluation

3. Traditional ML Approach

Data Preprocessing Pipeline

Step 1: Text Cleaning

- Convert to lowercase
- Remove special characters and digits
- Remove extra whitespace

Step 2: Advanced NLP Processing

- **Tokenization:** Break text into individual words
- **Stopword Removal:** Filter common non-informative words
- **Lemmatization:** Convert words to base form (running → run)

Step 3: Vectorization

- **CountVectorizer:** Word frequency counting

- **TF-IDF Vectorizer:** Importance-weighted word frequency
- Features: 5,000 (unigrams + bigrams)
- Train/Test split: 80/20

Models Trained

1. Naive Bayes
2. Logistic Regression (class_weight='balanced')
3. Linear SVM (class_weight='balanced')
4. **Random Forest** (class_weight='balanced_subsample')
5. XGBoost (with sample weights)
6. Gradient Boosting
7. Extra Trees (class_weight='balanced')

Best Performance

Random Forest: ~80% accuracy

- Fast training and inference
- Interpretable features
- Good balance across all classes

4. Transformer Approach (HuggingFace)

What are Transformers?

- Revolutionary deep learning architecture
- Use **self-attention mechanisms** to process text
- Learn contextual relationships from raw text
- Pre-trained on massive text corpora

Key Advantages

✓ **Contextual understanding:** Words have different meanings based on context
✓ **Automatic feature learning:** No manual engineering needed
✓ **Transfer learning:** Pre-trained models fine-tuned for specific tasks
✓ **Bidirectional processing:** Read text in both directions

Models Evaluated

1. **BERT** (bert-base-uncased) - Pioneering transformer
2. **RoBERTa** (roberta-base) - Optimized BERT variant
3. **DistilBERT** (distilbert-base-uncased) - 60% smaller, 97% performance
4. **ELECTRA** (google/electra-base-discriminator) - Efficient pre-training

Preprocessing Pipeline

1.1 Data Cleaning & Tokenization

- HuggingFace tokenizers (WordPiece, BPE)
- Special tokens: [CLS], [SEP], [PAD]
- Max sequence length: 256 tokens

1.2 Data Encoding

- Convert tokens to numerical IDs
- Create attention masks for variable-length sequences
- Handle padding and truncation

Baseline Results (No Fine-tuning)

Model	Accuracy	Precision	Recall	F1-Score
BERT	73.8%	75.9%	73.8%	74.7%
RoBERTa	73.0%	74.3%	73.0%	73.4%
DistilBERT	**78.0%**	**70.2%**	**78.0%**	**73.7%**

Best Baseline: DistilBERT (78% accuracy)

Fine-Tuning Implementation

- External module: `fine-tuning.py`
- Custom `SentimentDataset` class
- HuggingFace `Trainer` with `TrainingArguments`
- Early stopping callback
- Class-weighted loss function

5. Results & Comparison

Performance Summary

Approach	Best Model	Accuracy	Speed	Interpretability
Traditional ML	Random Forest	~80%	⚡ Fast	✓ High
Transformer (Baseline)	DistilBERT	78%	▯ Slower	✗ Low

Traditional ML Results

Random Forest (TF-IDF)

- **Accuracy:** ~80%
- **Training time:** Minutes
- **Inference:** Milliseconds per review
- **Features:** 5,000 interpretable n-grams

Transformer Results

DistilBERT (Baseline)

- **Accuracy:** 78%
- **Training time:** Hours (with fine-tuning)
- **Inference:** ~100ms per review
- **Features:** 768-dim contextual embeddings

Evaluation Metrics

Both approaches evaluated using:

- **Accuracy:** Overall correctness
- **Precision:** True positives / All predicted positives
- **Recall:** True positives / All actual positives
- **F1-Score:** Harmonic mean (macro & weighted)
- **Confusion Matrix:** Per-class performance

Trade-offs Analysis

Traditional ML Advantages:
✓ Fast training and inference
✓ Interpretable features (word importance)
✓ Lower computational requirements
✓ Easy deployment in production

Transformer Advantages:
✓ Contextual understanding
✓ Better with complex language patterns
✓ Pre-trained knowledge from massive corpora
✓ State-of-the-art performance (with fine-tuning)

6. Conclusions & Recommendations

Key Findings

✅ **Both approaches successfully meet project requirements**

- Traditional ML: 7 models trained and evaluated
- Transformers: 4 models evaluated (baseline + fine-tuning setup)

✅ **Random Forest delivers excellent production performance**

- 80% accuracy with fast inference
- Interpretable and easy to maintain

✅ **DistilBERT provides strong baseline**

- 78% accuracy without fine-tuning
- Room for improvement with fine-tuning

✅ **Class imbalance successfully addressed**

- Class weights and mild oversampling
- Macro-F1 tracking minority class performance

Project Requirements Fulfilled

README.md **Requirements:**

✓ **Traditional ML Approach**

- Data preprocessing (cleaning, tokenization, lemmatization)
- Vectorization (Count & TF-IDF)
- Multiple models trained
- Comprehensive evaluation metrics

✓ **Transformer Approach**

- Data cleaning and tokenization (HuggingFace)
- Data encoding (numerical IDs)
- Model selection (BERT, RoBERTa, DistilBERT, ELECTRA)
- Baseline evaluation

- Fine-tuning framework (external module)

✓ **Deliverables**

- Reproducible Jupyter notebook
- Documentation and analysis
- Presentation materials

Recommendations

For Production Deployment:

1. **Start with Random Forest** for speed and interpretability
2. **Consider DistilBERT** for complex reviews needing context
3. **Use ensemble approach** combining both methods

For Future Improvements:

1. Fine-tune transformers on full dataset
2. Implement threshold tuning for Neutral class
3. Explore BERT/RoBERTa for maximum accuracy
4. Build dashboard for real-time monitoring

Architecture Decision:

- **High-volume, real-time:** Traditional ML (Random Forest)
- **Complex analysis, batch processing:** Transformers (DistilBERT+)
- **Hybrid system:** ML for filtering, Transformers for edge cases

Thank You!

Questions?

Project Demonstrates:

- Comprehensive NLP pipeline
- Traditional ML vs Deep Learning comparison
- Production-ready implementations
- Proper evaluation methodology

Key Takeaway: Both approaches have their place - choose based on your specific requirements for speed, accuracy, and interpretability.