NLP Automated Customer Reviews: Comparative Analysis Report

Executive Summary

This project developed and compared natural language processing (NLP) models for automated sentiment classification of customer reviews. We evaluated **traditional machine learning approaches** against **modern transformer-based models**, analyzing **55,000 customer reviews** from multiple sources to determine the most effective solution for automated sentiment analysis.

Key Findings

- **Random Forest with TF-IDF** achieved the highest overall accuracy at **80.3%**, outperforming all transformer models
- **DistilBERT** was the best-performing transformer model with **78.0%** accuracy
- Traditional ML models demonstrated superior performance, faster training, and more efficient deployment characteristics
- The neutral sentiment class presented challenges across all models due to class imbalance

---

1. Project Overview and Methodology

1.1 Objective

Develop an automated system to classify customer reviews into three sentiment categories:

- **Negative:** Ratings 1-3
- **Neutral:** Rating 4
- **Positive:** Rating 5

1.2 Data Collection and Processing

- **Total Reviews:** 55,000 customer reviews
- **Data Sources:**
- HuggingFace IMDB Dataset: 25,000 movie reviews
- Local Amazon Dataset: 30,000 product reviews
- **Test Set Distribution:**

- Positive: 6,590 reviews
- Negative: 2,988 reviews
- Neutral: 1,345 reviews

Preprocessing Steps:

1. Text cleaning and normalization
2. Tokenization
3. Lemmatization
4. Stop word removal
5. Vectorization (Count Vectorizer and TF-IDF)

1.3 Methodology

We implemented a **dual-approach comparison**:

1. **Traditional ML Pipeline:** Feature engineering with Count/TF-IDF vectorization + classical algorithms
2. **Transformer Pipeline:** Pre-trained models (BERT, RoBERTa, DistilBERT, ELECTRA) evaluated as baselines

Class Imbalance Handling:

- Applied class weighting (1.15x weight for neutral class).
- Mild oversampling using Random Oversampler
- Sample weighting for XGBoost

Why this setup:

- Class weights shift the decision boundary toward minority classes with minimal risk of overfitting.
- Mild over-sampling gives Neutral more representation without over-amplifying noise (we avoid fully balancing to Positive).
    - By not fully balancing to Positive, we avoid introducing too much synthetic data/noise for the majority class.
- For transformers, we use a class-weighted CrossEntropyLoss in fine-tuning (WeightedTrainer) instead of resampling.

Scores:

- We are looking at <u>F1-score and recall</u> for the evaluation of our models, since it is the best approach for a class weight trained model (ML or Transformer).

---

2. Traditional Machine Learning Results

2.1 Complete Model Performance Comparison

Count Vectorizer Results

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	79.9%	79.5%	79.9%	78.7%
SVM	72.9%	76.7%	72.9%	74.3%
Gradient Boosting	75.6%	75.2%	75.6%	74.8%
Logistic Regression	69.6%	77.2%	69.6%	71.8%
XGBoost	57.9%	77.8%	57.9%	60.7%
Naive Bayes	51.3%	75.9%	51.3%	53.0%
Extra Trees	37.2%	71.3%	37.2%	28.0%

TF-IDF Vectorizer Results

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	**80.3%**	**79.8%**	**80.3%**	**79.3%**
Gradient Boosting	76.2%	76.0%	76.2%	75.5%
SVM	74.0%	78.1%	74.0%	75.4%
Logistic Regression	70.3%	78.9%	70.3%	72.5%
XGBoost	61.7%	78.1%	61.7%	64.4%
Naive Bayes	59.2%	72.9%	59.2%	61.4%
Extra Trees	36.9%	70.5%	36.9%	26.4%

2.2 Best Performing Model: Random Forest with TF-IDF

Overall Metrics:

- **Accuracy:** 80.3%
- **Precision:** 79.8%
- **Recall:** 80.3%
- **F1-Score:** 79.3%
- **Training Time:** 231.66 seconds

Per-Class Performance:

Class	Precision	Recall	F1-Score	Support
Negative	83.8%	76.0%	79.7%	2,988
Neutral	68.8%	39.1%	49.9%	1,345
Positive	80.3%	90.7%	85.2%	6,590

Confusion Matrix:

	Predicted Negative	Predicted Neutral	Predicted Positive
Negative	2,271	59	658
Neutral	6	526	813
Positive	434	180	5,976

Key Observations:

- Excellent performance on positive and negative sentiment detection
- Struggled with neutral sentiment classification due to class imbalance
- Strong recall for positive class (90.7%)
- Neutral class achieved only 39.1% recall despite class weighting strategies

Best model chosen:

- I used Grid Search for finding the best hyper-parameters for XGBoost and RandomForest models, both with TF-IDF vectorization.
    - They work well with multi-class prediction, since they are Ensemble methods. Also, RF works well with over-adjustment.

Some reasons for the Neutral score in the confusion matrix:

Random Forest:

- 4 and 5 (Neutral vs Positive) are often semantically close.
- After the sampling strategy (neutral gets closer to negative), the model is still biased to positive.

XGBoost:

- Imbalanced set, too many positives form IMDB (binary dataset)
- Weight biased towards neutral values.

	Negative	Neutral	Positive
Negative	2256	75	657
Neutral	14	634	697
Positive	442	436	5712

Random Forest

	Negative	Neutral	Positive
Negative	2279	453	256
Neutral	5	1313	27
Positive	772	3789	2029

XGBoost

---

3. Transformer Model Results

3.1 Baseline Performance (Pre-trained Models - No Fine-tuning)

Model	Architecture	Accuracy	Precision	Recall	F1-Score
DistilBERT	**Distilled BERT**	**78.0%**	**70.3%**	**78.0%**	**73.7%**
BERT	Base Uncased	73.8%	75.9%	73.8%	74.7%

RoBERTa	Twitter Optimized	73.0%	74.3%	73.0%	73.4%
ELECTRA	Token Replacement	27.2%	7.4%	27.2%	11.6%

Model Selection Rationale:

DistilBERT was selected as the best transformer baseline due to:

- **Superior accuracy** (78.0%) among all transformers tested
- **Computational efficiency:** 60% smaller than BERT
- **Faster inference:** 60% faster than full BERT
- **Good balance** between speed and accuracy
- **Easier deployment** in production environments

Why have I added ELECTRA:
- Newer architecture, but less stablished.
- Better performance and sample-efficient than BERT, since it learns from all tokens.

3.2 Best Performing Transformer: DistilBERT Baseline

Overall Metrics:

- **Accuracy:** 78.0%
- **Precision:** 70.3%
- **Recall:** 78.0%
- **F1-Score:** 73.7%

Per-Class Performance:

Class	Precision	Recall	F1-Score	Support
Negative	67.4%	89.7%	77.0%	136
Neutral	0.0%	0.0%	0.0%	55
Positive	84.0%	86.7%	85.4%	309

Confusion Matrix:

	Predicted Negative	Predicted Neutral	Predicted Positive
Negative	122	0	14
Neutral	18	0	37
Positive	41	0	268

Key Observations:

- **Complete failure** to predict neutral class (0% precision and recall)
- Strong performance on positive and negative classes
- Model collapsed neutral predictions into the majority classes
- Pre-trained sentiment models not optimized for 3-class sentiment task

Why is neutral missing? Some possible causes:

- Many pre-trained "sentiment-analysis" checkpoints (e.g., SST-2 distilBERT) are binary and will never output a Neutral class.
    - As a reminder, I address imbalance with class-weighted and RandomOverSampler
- MOST PROBABLY: IMDB dataset is binary, so when mapped it only fixated in two values.

Pros and cons of using a combined dataset:

- Better prediction for Positive and Negative, but neutral get lost.
- Neutral can be learnt better with a traditional LM.

---

4. Comprehensive Comparative Analysis

4.1 Performance Comparison

Approach	Best Model	Accuracy	F1-Score	Precision	Recall
Traditional ML	**RF + TF-IDF**	**80.3%**	**79.3%**	**79.8%**	**80.3%**
Transformer	DistilBERT Baseline	78.0%	73.7%	70.3%	78.0%

Performance Gap: Traditional ML outperformed transformers by **2.3% in accuracy** and **5.6% in F1-score**.

4.2 Speed and Efficiency Comparison

Model	Training Time	Inference Speed	Computational Requirements
RF + TF-IDF	231.66s	Very Fast	CPU sufficient
DistilBERT	Slow	Moderate	GPU recommended
BERT/RoBERTa	Very Slow	Slow	GPU required

4.3 Per-Class Performance Comparison

Negative Class:

- Random Forest: 83.8% precision, 76.0% recall
- DistilBERT: 67.4% precision, 89.7% recall

Neutral Class:

- Random Forest: 68.8% precision, 39.1% recall
- DistilBERT: 0.0% precision, 0.0% recall

Positive Class:

- Random Forest: 80.3% precision, 90.7% recall
- DistilBERT: 84.0% precision, 86.7% recall

4.4 Key Findings

1. **Traditional ML superiority:** Random Forest with TF-IDF achieved the best overall performance
2. **Neutral class challenge:** Both approaches struggled with neutral sentiment, but transformers failed completely
3. **Speed advantage:** Traditional ML trains and infers significantly faster
4. **Resource efficiency:** Traditional ML requires only CPU, making deployment easier
5. **Model interpretability:** Traditional ML offers better feature importance analysis

---

5. Business Implications and Recommendations

5.1 Primary Recommendation

Deploy Random Forest with TF-IDF for production use based on:

Higher Accuracy: 80.3% vs 78.0%
Better F1-Score: 79.3% vs 73.7%
Fast Training: Suitable for regular retraining
Real-time Inference: Very fast predictions
Lower Computational Requirements: No GPU needed
Model Interpretability: Feature importance available
Easy Deployment: Standard scikit-learn compatibility
Cost-Effective: Lower infrastructure costs

5.2 Use Cases for Traditional ML (Recommended)

- Real-time sentiment analysis systems
- Resource-constrained environments
- High-throughput batch processing
- Applications requiring model interpretability
- Budget-conscious implementations

- Quick deployment requirements

5.3 When to Consider Transformer Models

Transformer models may be preferable when:

- Fine-tuning with large domain-specific datasets is feasible
- Deep language understanding is critical
- Abundant computational resources (GPUs) are available
- Latest state-of-the-art performance is mandatory
- Budget allows for higher infrastructure costs

Note: Without fine-tuning, transformers underperformed traditional ML in this task.

---

6. Challenges and Limitations

6.1 Class Imbalance Issues

Test Set Distribution:

- Positive: 60.9% (6,590 reviews)
- Negative: 27.6% (2,988 reviews)
- Neutral: 12.4% (1,345 reviews)

Impact:

- Neutral class significantly underrepresented
- All models struggled with neutral predictions
- DistilBERT completely failed to predict neutral class
- Random Forest achieved only 39.1% neutral recall despite mitigation strategies

Mitigation Attempts:

- Class weighting (1.15x for neutral)
- Random oversampling
- Sample weighting for XGBoost
- Results: Partial improvement but challenge persists

6.2 Transformer Limitations Observed

- Pre-trained models optimized for binary sentiment (positive/negative)
- Neutral class completely ignored by DistilBERT
- Fine-tuning required for 3-class sentiment task
- High computational requirements without performance gain
- Slow inference speed compared to traditional ML

---

7. Future Improvements

7.1 Some possible improvements

1. **Enhanced Class Balancing:**
    1. Advanced oversampling techniques (SMOTE, ADASYN)
    2. More aggressive synthetic data generation for neutral class
    3. Cost-sensitive learning with dynamic weights
2. **Feature Engineering:**
    1. N-gram analysis (bigrams, trigrams)
    2. Sentiment lexicons integration
    3. Part-of-speech tagging features
3. **Ensemble Methods:**
    1. Combine Random Forest and SVM predictions
    2. Stacking with meta-learner
    3. Voting classifier with multiple models
4. Replace the alternative dataset:
    1. I would consider an alternative dataset as SemEval.

---

8. Conclusion

This comprehensive study demonstrates that **traditional machine learning approaches**, specifically **Random Forest with TF-IDF vectorization**, outperform pre-trained transformer models for sentiment classification of customer reviews.

Final Results Summary

Metric	Random Forest + TF-IDF	DistilBERT Baseline	Winner
Accuracy	80.3%	78.0%	Traditional ML
F1-Score	79.3%	73.7%	Traditional ML
Speed	Very Fast	Moderate	Traditional ML
Resources	CPU Only	GPU Preferred	Traditional ML
Deployment	Easy	Complex	Traditional ML

Full Dashboard

A full dashboard file is available. You can open it locally using "python3 -m http.server 8000 --directory Dashboard".
The dashboard is a static HTML/JavaScript page (index.html) that runs in the browser and loads JSON summaries.

An alternative dashboard is available in the Jupyter Notebook file, using Plotly.

Failed Implementation of fine-tuning

During the project, the fine-tuning failed, for some unknown reason. At first, it was a compatibility problem with Python 3.11. Then, it was a problem downloading models or weights from Hugging-Face. Later, it was a problem with Visual Studio Code where the cells didn't load. I stopped the implementation of this feature, but I added as an file .py.

Generative AI

A method for using generative AI for summarizing reviews is available, but not fully implemented.

Final Recommendation

Implement Random Forest with TF-IDF for immediate production deployment based on:

- Superior accuracy and F1-score
- Significantly faster training and inference
- Lower computational and infrastructure costs
- Easier deployment and maintenance
- Better model interpretability

Parallel Development Strategy:

- Continue exploring transformer fine-tuning as a parallel development track
- Investigate hybrid approaches combining traditional and deep learning
- Address neutral class imbalance through advanced sampling techniques
- Monitor emerging NLP techniques for future improvements

The combination of high performance, operational efficiency, and cost-effectiveness makes traditional ML the clear choice for this sentiment analysis application.

Appendix: Technical Details

A. Training Configuration

Traditional ML:

- Random Forest: 400 estimators, balanced subsample weighting
- SVM: Linear kernel, balanced class weights
- Logistic Regression: L-BFGS solver, multinomial classification
- Training data: Balanced with Random Oversampler

Transformers:

- All models: Pre-trained, no fine-tuning
- Max sequence length: 512 tokens
- Batch processing: 1 sample at a time (baseline evaluation)
- Device: MPS (Apple Silicon M4)

B. Evaluation Metrics

- **Accuracy:** Overall correct predictions / total predictions
- **Precision:** True positives / (True positives + False positives)
- **Recall:** True positives / (True positives + False negatives)
- **F1-Score:** Harmonic mean of precision and recall
- **Macro F1:** Average F1-score across all classes (equal weight)
- **Weighted F1:** F1-score weighted by class support

C. Hardware and Software Environment

- **Hardware:** MacBook Air M4
- **OS:** macOS
- **Python:** 3.10
- **Key Libraries:**
- scikit-learn: Traditional ML models
- transformers: Hugging Face transformer models
- pandas, numpy: Data processing
- imbalanced-learn: Class imbalance handling

Project: NLP Automated Customer Reviews Analysis
Status: Production Recommendation Ready