RoboReviews:

Smart Insights from Customer Voice

Presented by:
Michael A. Libio
Tejal D. Bhatti

Pitch


Ever feel lost in a sea of reviews? We turn that chaos into clarity.
RoboReviews:
Your AI assistant for instant sentiment and smart product organization
Get ready to see how AI brings clarity, organization, and actionable insights to product feedback.

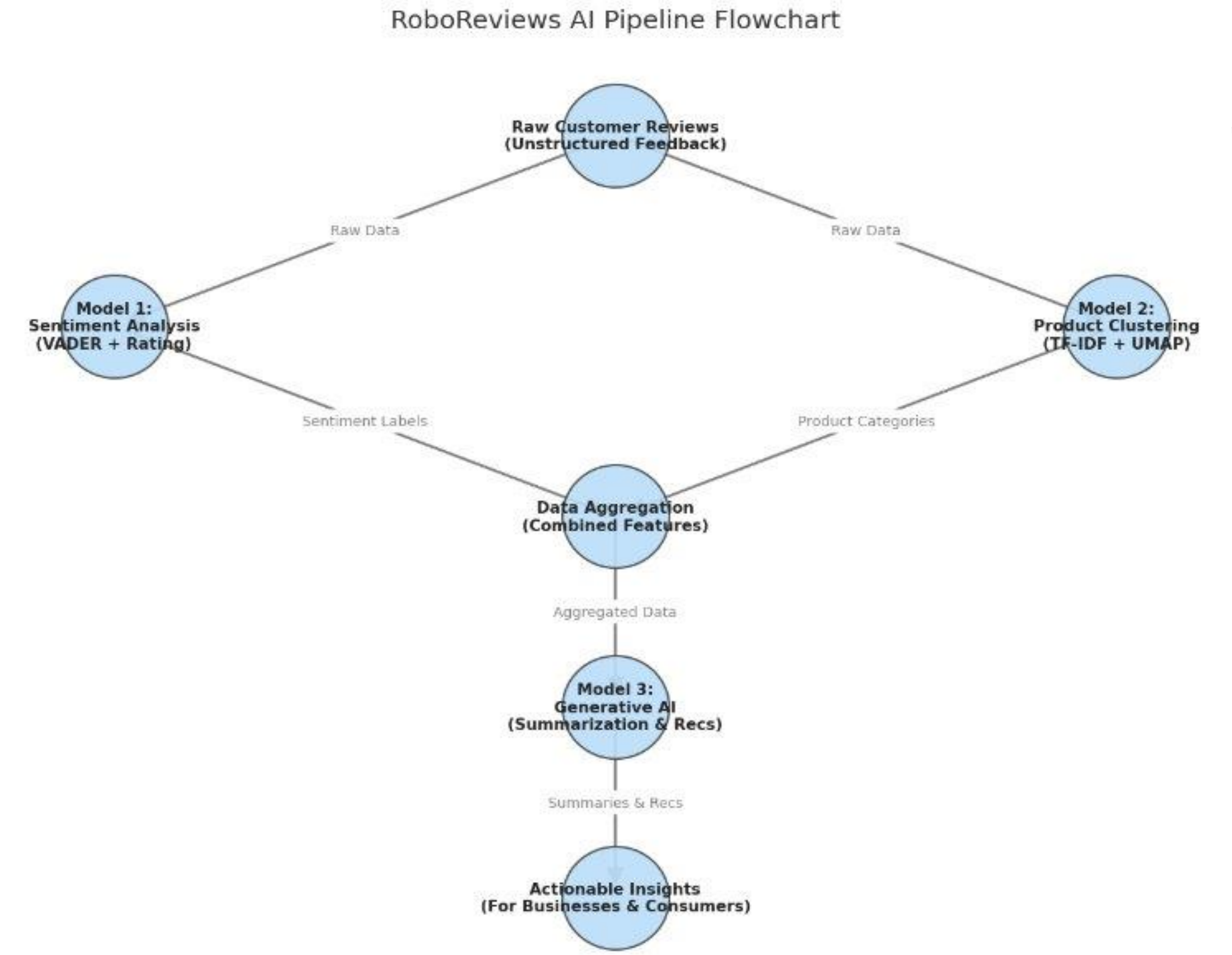
Introduction

Real-World Problem: "Customers face **'review fatigue'** when trying to purchase products online. Shifting through **thousands of reviews is time-consuming and overwhelming**, making it difficult to find critical insights and make informed decisions."

Our Solution: RoboReviews

An intelligent AI system for sentiment analysis, product categorization, and automated recommendations, utilizing Generative AI to summarize reviews.

Roboreview AI Pipeline



Model 1

- **Classify customer reviews into positive, negative, or neutral** categories to help the company improve its products and services.

- **Source:** Amazon Product Reviews Dataset (`1429_1.csv`).
- **Size: 34,660 reviews**.
- **Key Data:** `reviews.text` (review content) & `reviews.rating` (star rating).

Model 1: Dataset Challenges & Solutions

	proportion
sentiment	
positive	92.15%
neutral	5.01%
negative	2.85%

dtype: object

- **Challenge 1**: High Class Imbalance
- **Solution**: Balanced Model Approach
- **Why Sampling Not Ideal**:
  - Under sampling: Reduced data to ~1500 samples, losing critical information and generalization ability.
  - Oversampling: Risk of overfitting on duplicated synthetic patterns, especially with extreme 90% imbalance.
- **Our Strategy:** "Used a balanced SVC model with `class_weight='balanced'` on the full dataset to learn from all data while internally weighting underrepresented classes."

Model 1: Dataset Challenges & Solutions

- **Challenge 2:** Label Mismatches (Rating vs. Text)
- **Problem:** Star ratings don't always align with the actual sentiment in the text .
- **Solution:** VADER Sentiment

```
Review Example: "Not easy for elderly users cease of ads that pop up."
↓
Original Rating (4.0 Stars)
↓
Initial Sentiment (Rating-based: Positive)
↓
VADER Analysis of Review Text (Identifies "not easy" and "ads that pop up" as negative)
↓
Reconciliation & Override
↓
Final Label (VADER-based: Negative)
```

Model 1: Sentiment Analysis Workflow

```
Raw Reviews (Text & Ratings)
↓
1. Create Initial Sentiment Labels from Ratings (4-5 stars: Positive, etc.)
↓
2. VADER Sentiment Analysis & Label Reconciliation (Correcting rating-text mismatches)
↓
3. Feature Extraction: TF-IDF Vectorization (Text -> Numerical Features, max_features=5000)
↓
4. Model Training: Support Vector Classifier (SVC)
(Key Params: kernel='linear', probability=True, class_weight='balanced')
↓
Final Output: product_reviews_sentiment_and_confidence.csv
(Contains: Review Text, Predicted Sentiment, Confidence Scores)
```

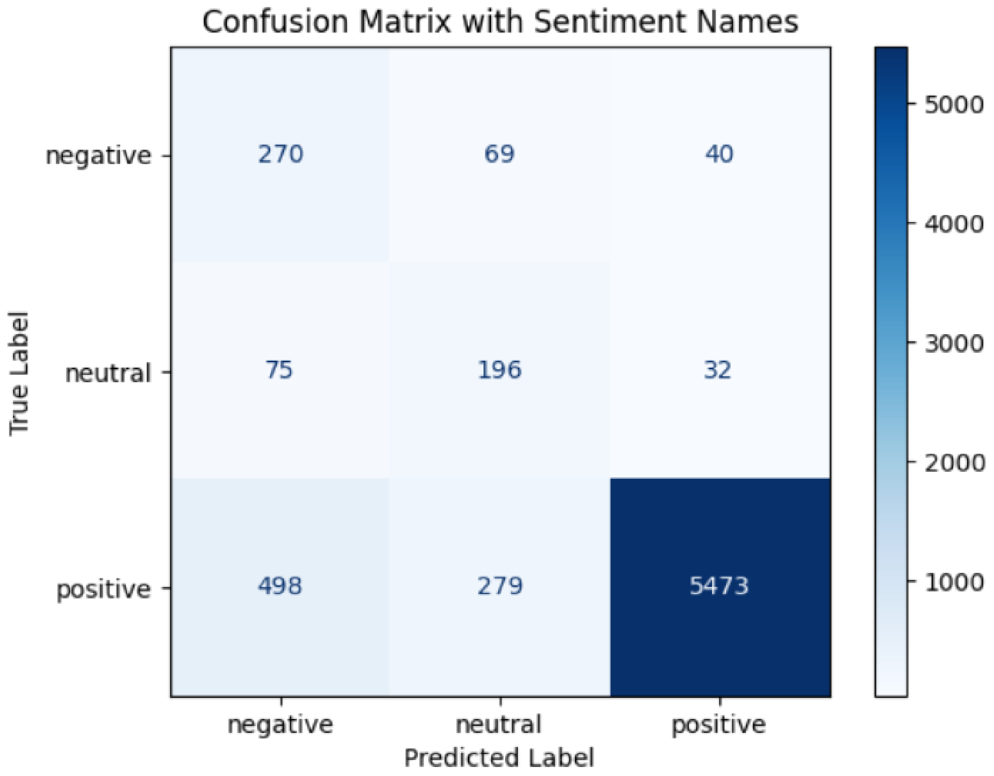
Model 1: Evaluation

The TF-IDF + SVC model achieved:

- **Accuracy: 85.68%**
- **Classification Report:**

Classification Report: precision recall f1-score support

negative	0.32	0.71	0.44	379
neutral	0.36	0.65	0.46	303
positive	0.99	0.88	0.93	6250
accuracy			0.86	6932
macro avg	0.56	0.74	0.61	6932
weighted avg	0.92	0.86	0.88	6932



Model 2

- **Objective:** To group Amazon products into 4-6 meaningful categories using advanced NLP techniques to simplify organization and enhance insights.
- **Input Data:** Aggregated product data, including `name` and `categories` from Amazon reviews.
- **Output** : all Products with category classifed into 5 clusters

Model 2: Product Clustering Workflow

Raw Product Data (Name & Categories)

↓

1. Data Preparation: Handle Duplicates/Missing Values , Extract Most Specific Category

(e.g., "Electronics, Tablets, Amazon Devices, Fire Tablets" -> "Fire Tablets")

- Combine (Name + Most Specific Category) -> text_for_clustering

↓

2. Text Embedding: Model: 'all-MiniLM-L6-v2' Sentence Transformer

(Convert text_for_clustering -> Dense Vector Embeddings)

↓

3. Clustering Algorithms: KMeans,Agglomerative Clustering, HDBSCAN

↓

Final Output: category_cluster_with_id.csv

(Products with assigned cluster)

Model 3: Use Generative AI to summarize reviews into an article which recommends the top products for each category

Problem: Overwhelming volume of customer reviews.

Challenge: Difficult to quickly extract insights and identify top products.

Initial LLM Issues: Inconsistent quality, generic summaries, varied formatting with few-shot prompting.

Solution Needed: A system for consistent, high-quality, professional, and structured product recommendation articles from raw review data.

Model 3: What we did and why

```python
contexts = {
    'Fire Tablets & Echo Speakers': 'smart home and family tech expert',
    'E-Readers & Kindle Devices': 'digital reading and e-ink specialist',
    'Charging & Accessories
    'Kindle Cases & Covers'
    'Fire TV & Streaming De
}
```

```python
        # Section 1: Opening hook (small prompt)
        opening_prompt = f"""Write 2-3 sentences introducing {category} based on these \
customer feedback trends. Do not include phrases like "Okay" or "Here is". \
Start directly with the content.

POSITIVE: {insights['positive_examples']}
NEGATIVE: {insights['negative_examples']}

Write like a tech blogger setting the scene for readers."""
        review_sections['opening'] = self.generate_single_prompt(opening_prompt)

        # Section 2: Strengths analysis (focused prompt)
        strengths_prompt = f"""Based on positive customer feedback: \
{insights['positive_examples']}

Write one paragraph explaining what's working well in {category}. \
Mention specific benefits customers report. Be conversational and specific. \
Do not start with "Okay" or similar phrases."""
        review_sections['strengths'] = self.generate_single_prompt(strengths_prompt)

        # Section 3: Concerns analysis (focused prompt)
        concerns_prompt = f"""Based on negative customer feedback: \
{insights['negative_examples']}

Write one paragraph about the main problems customers face with {category}. \
Be specific about issues and write like you're warning readers. \
Do not start with "Okay" or similar phrases."""
        review_sections['concerns'] = self.generate_single_prompt(concerns_prompt)

        # Section 4: Final recommendation (synthesis prompt)
        recommendation_prompt = f"""For {category}, considering both the positives and \
negatives from customer reviews, write 2-3 sentences advising who should buy and \
who should avoid these products. Be direct and helpful. \
Do not start with "Okay" or similar phrases."""
        review_sections['recommendation'] = self.generate_single_prompt(recommendation_prompt)
```

Phase 1: Few-Shot Prompting (Initial Approach)

What we tried: We initially used generic instruction-tuned models (e.g., Gemma-2B, Mistral, Qwen) with in-context learning through carefully crafted prompts. Dynamic prompt construction integrated review data directly.

Why: This approach offered flexibility and immediate deployment without requiring model training. The prompt logic was also interpretable.

Limitations: inconsistent output quality, significant variance by model and prompt, context window limitations, and high prompt engineering overhead. Output quality was variable and often lacked the desired professional tone and consistent structure.

Model 3: What we did and why

Phase 2: Multi-Model Comparison

What we tried: We systematically evaluated multiple models (Gemma-2B, Mistral, Qwen) using identical prompts to benchmark performance and standardize our pipeline.

Why: This helped us understand the quality hierarchy (7B models > 2B models for complex reasoning), the speed vs. quality trade-off, and model-specific behaviors.

This phase highlighted the need for greater consistency, task specialization, efficiency gains, and format standardization that generic models couldn't consistently provide.

Model 3: What we did and why

- Base model: Qwen & Gemma 3
- Training data: 173 high quality data (synthetic via Gemini)
- Training: LoRA Fine-tuning was used for 3 epochs with a batch size of 1, learning rate of 2×10^{-4}


```
{
  "instruction": "Create a product recommendation guide for Fire TV & Streaming Devices based on customer reviews and rat
  "response": "### **Navigating the Streaming Jungle: What Customers Are Saying About Fire TV & Beyond**\n\nExtensive anal
  "category": "Fire TV & Streaming Devices"
},
{
  "instruction": "Create a product recommendation guide for Fire TV & Streaming Devices based on customer reviews and rat
  "response": "### **Decoding the Stream: An Honest Look at Fire TV Performance**\n\nSignificant time evaluating the real-
  "category": "Fire TV & Streaming Devices"
},
{
  "instruction": "Create a product recommendation guide for Fire TV & Streaming Devices based on customer reviews and rat
  "response": "### **Fire TV Insights: Unpacking the Customer Experience**\n\nIn the ever-evolving world of streaming, a c
  "category": "Fire TV & Streaming Devices"
},
{
  "instruction": "Create a product recommendation guide for Fire TV & Streaming Devices based on customer reviews and rat
  "response": "### **Beyond the Hype: A User-Centric Review of Fire TV Devices**\n\nRegular analysis of tech trends reveal
  "category": "Fire TV & Streaming Devices"
},
```


Phase 3: Fine-Tuning Specialization

What we tried: decided to fine-tune our specialized model

Why: the need for consistent output quality, optimization for product recommendation generation, efficiency gains (reduced prompt engineering), and format standardization.

Challenges: like MPS numerical stability, memory optimization, proper label masking, and seamless model integration

Model 3: Some metrics (what we observed)

Metric	Few-Shot Prompting	Fine-Tuned Model
Consistency	Variable (60-80%)	High (95%+)
Speed	Slower (large prompts)	Faster (simple prompt)
Memory Usage	High (context + model)	Lower (efficient inference)
Quality	Model-dependent	Low (model handles format)
Maintenance	High (prompt engineering)	Lower (efficient inference)
Customization	Limited (prompt constraints)	High (learned preferences)

Takeaways

LLM ~~journey~~ rabbit hole

- Work backwards (from goal to implementation)
- Understand the tradeoffs between options
- Commercial LLMs, based on price and speed (and requirements)

Demo

Product Review Insights Dashboa
Overall Insights
Total Products Analyzed:
Total Reviews Processed:
Average Rating Across All Products:
Sentiment Distribution:
Charging & Accessories
Customer Insights
The Charging & Accessories Landscape: A Comprehensive Look
Feedback
What's Working Really Well:
The Red Flags to Watch:
My Recommendation:
The Red Flags to Watch:
My Recommendation:
What Customers Love
Excellent for family use and children
Good value proposition in the market
Performance issues mentioned in feedback
Top Recommendations
Customer Highlights:

Resource Efficiency Gains

Resource	Few-Shot	Fine-Tuned	Improvement
Prompt Length	800-1200 tokens	20-50 tokens	95% reduction
Inference Time	3-5 seconds	1-2 seconds	50% faster
Memory Usage	Variable	Consistent	Predictable
Context Utilization	60% prompt, 40% generation	10% prompt, 90% generation	2.25x more generation