

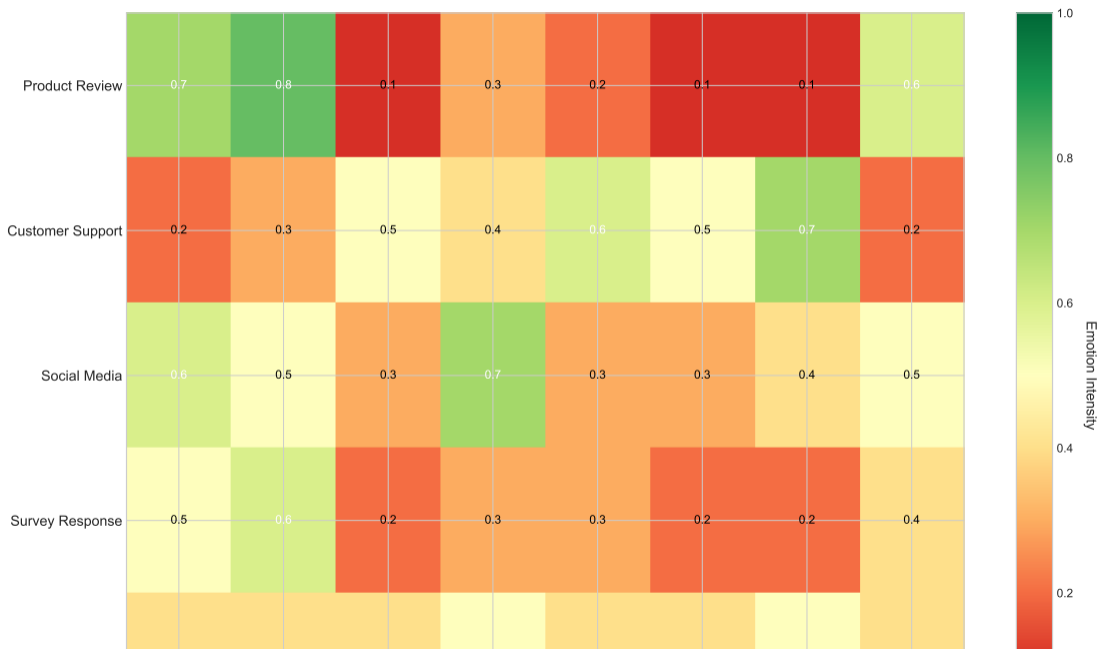
Machine Learning for Smarter Innovation

Week 3: NLP for Emotional Context
Understanding User Emotions Through Language

ML & Design Thinking Course

2025

Emotion Spectrum Across Different Contexts



Why Language Reveals What Users Really Feel

Traditional Analysis

- Keyword counting
- Manual categorization
- Surface-level insights
- Limited scale
- Misses context

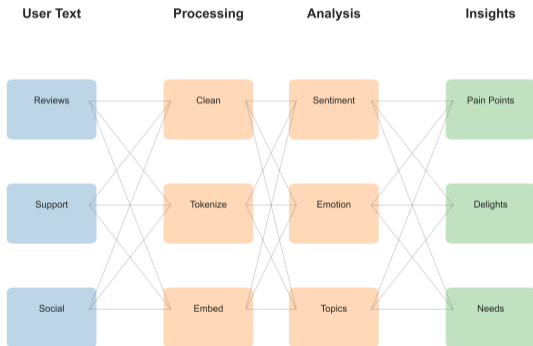
NLP-Powered Analysis

- Contextual understanding
- Emotion detection
- Sarcasm recognition
- Unlimited scale
- Deep insights

“I love waiting 2 hours for support!” → Negative (sarcasm detected)

Every Word Tells a Story

From Language to Design Insights



What Language Reveals:

- Frustration points
- Delight moments
- Unmet expectations
- Hidden needs
- Emotional journey

Design Insights:

- Priority pain points
- Feature requests
- User segments
- Experience gaps

Same Words, Different Meanings

Context Changes Everything

<p>"This is sick!"</p> <p><i>Gaming Community</i></p> <p>Positive</p>	<p>"This is sick!"</p> <p><i>Healthcare Forum</i></p> <p>Negative</p>	<p>"It's fine..."</p> <p><i>After Complaint</i></p> <p>Negative</p>
<p>"It's fine!"</p> <p><i>First Experience</i></p> <p>Positive</p>	<p>"Whatever"</p> <p><i>Teen Response</i></p> <p>Neutral</p>	<p>"Whatever"</p> <p><i>Support Chat</i></p> <p>Negative</p>

"This is sick!"

- Gaming: Positive

"It's fine"

- After complaint: Negative

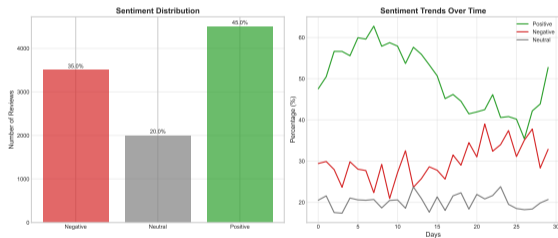
"Interesting..."

- Academic: Positive

From Polarity to Emotional Spectrum

Sentiment Analysis

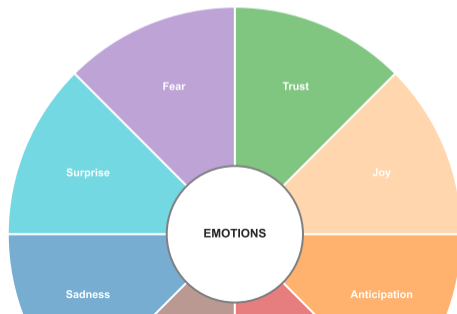
- Positive / Negative / Neutral
- Overall polarity
- Simple classification
- Good for trends



Emotion Analysis

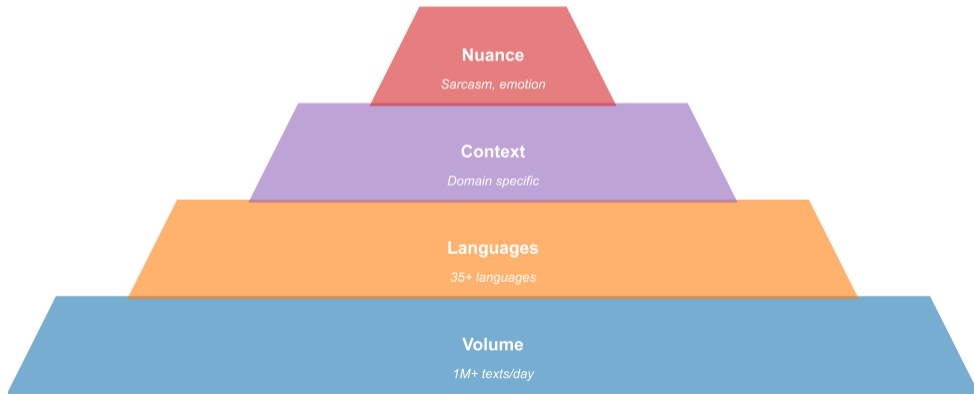
- Joy, Anger, Fear, Sadness, Surprise...
- Nuanced understanding
- Multi-class detection
- Better for design

Plutchik's Wheel of Emotions



Processing Millions While Preserving Meaning

The NLP Challenge: Scale Meets Nuance



What You'll Master

Technical Skills

- Text preprocessing pipelines
- BERT and transformers
- Sentiment classification
- Emotion detection
- Aspect-based analysis
- Model evaluation

Design Applications

- Voice of Customer analysis
- Emotional journey mapping
- Pain point identification
- Feature prioritization
- Persona refinement
- Experience optimization

Outcome: Transform text into actionable design insights at scale

Companies Using NLP for Better Design

Airbnb

- Review analysis
- Experience gaps
- Host coaching
- 23% better ratings

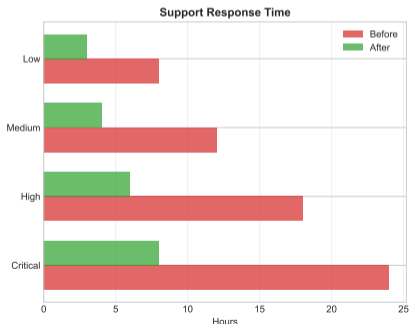
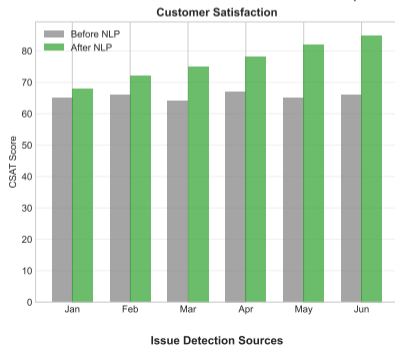
Spotify

- Mood detection
- Playlist curation
- User feedback
- 40% more engagement

Slack

- Support tickets
- Feature requests
- User frustration
- 50% faster resolution

NLP Impact on Business Metrics

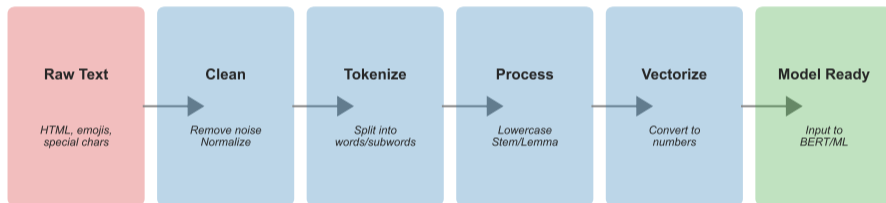


Part 2: Technical Deep Dive

From Text to Understanding

The Critical First Steps

Text Preprocessing Pipeline



<p>I LOVE this!!! ☹️</p>

I LOVE this!!!

['I', 'LOVE', 'this']

['I', 'love', 'this']

[0.2, 0.8, 0.3]

Tensor([...])

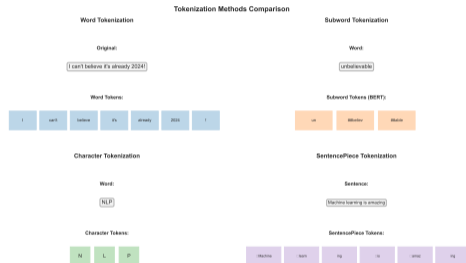
Breaking Down Language

Tokenization Methods:

- Word-level: "I love this" → ["I", "love", "this"]
- Subword: "unbelievable" → ["un", "believ", "able"]
- Character: "ML" → ["M", "L"]
- Byte-Pair Encoding (BPE)
- WordPiece (BERT)

Why It Matters:

- Handles out-of-vocabulary words
- Captures morphology
- Reduces vocabulary size



Capturing Meaning in Vectors

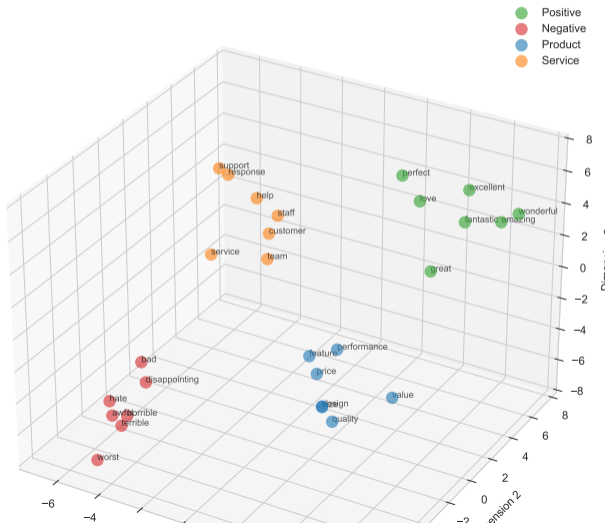
Evolution of Word Embeddings

From Sparse to Dense to Contextual



Similar Words, Nearby Vectors

Word Embeddings in 3D Space



Key Properties:

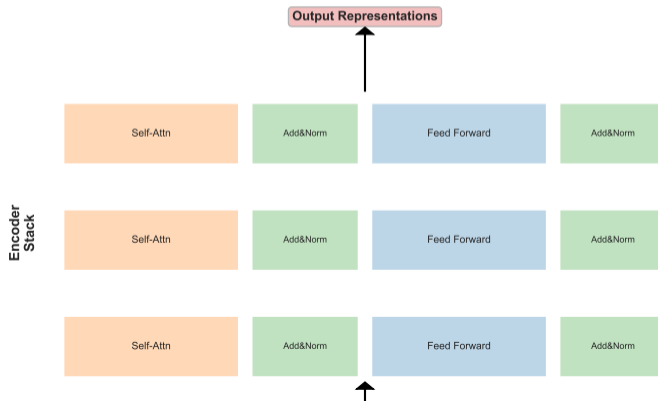
- Semantic relationships
- Analogies work: $\text{King} - \text{Man} + \text{Woman} = \text{Queen}$
- Distance = similarity
- Clustering by meaning

Popular Models:

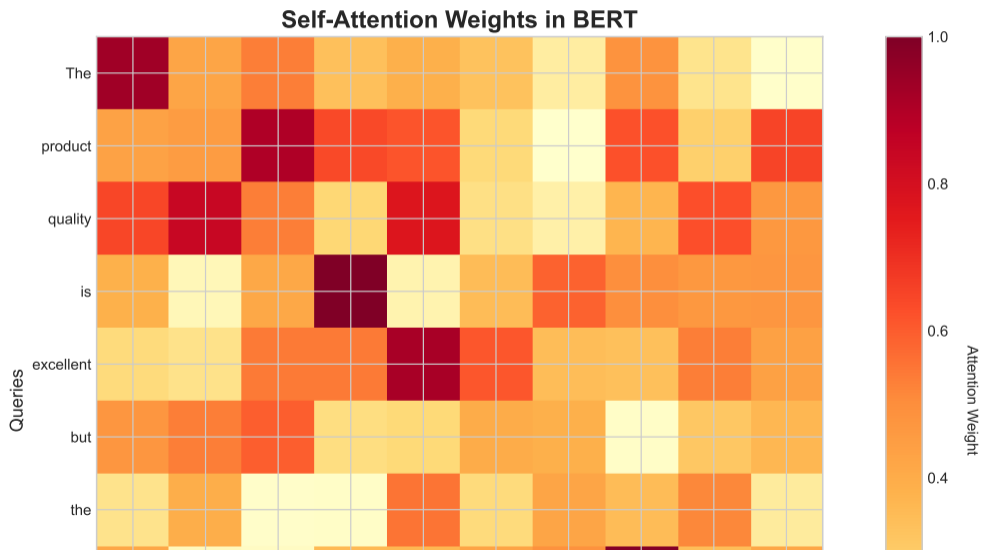
- Word2Vec (Google)
- GloVe (Stanford)
- FastText (Facebook)
- Contextual: ELMo, BERT

Attention is All You Need

Transformer Encoder Architecture



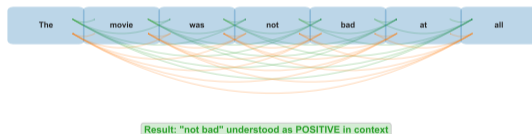
How Transformers “Pay Attention”



Reading Both Ways for Better Context

BERT: Bidirectional Understanding

Each word sees all other words simultaneously



BERT Innovation:

- Bidirectional context
- Masked language modeling
- Next sentence prediction
- Transfer learning

Example: "The bank is by the [MASK]"

- River → "bank" = shore
- Street → "bank" = financial

Pre-train Once, Fine-tune Many

BERT Pre-training Tasks

Masked Language Modeling

Original: The cat sat on the mat

Training: The [MASK] sat on the [MASK]

Predicted: The cat sat on the mat

Next Sentence Prediction

A: I love machine learning.

B: It's fascinating.

IsNext

A: The weather is nice.

B: Pizza is delicious.

NotNext

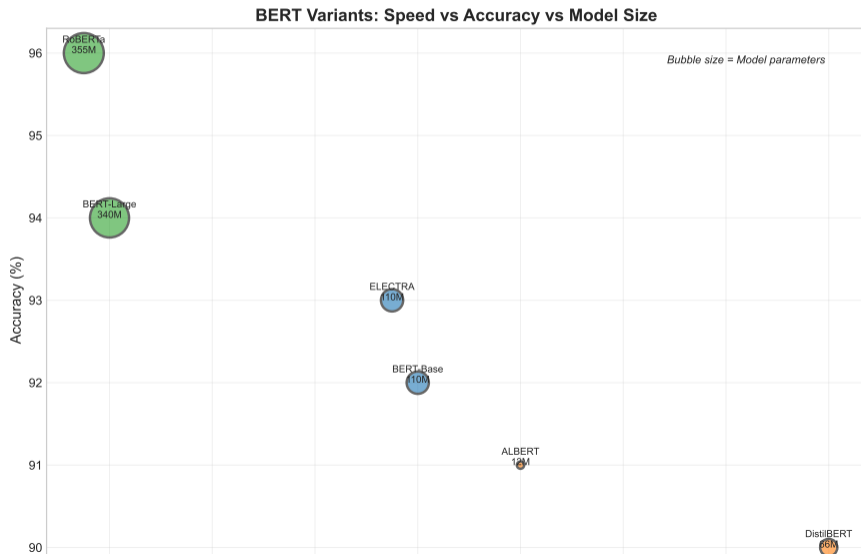
Pre-training (Google):

- 3.3 billion words
- Wikipedia + BookCorpus

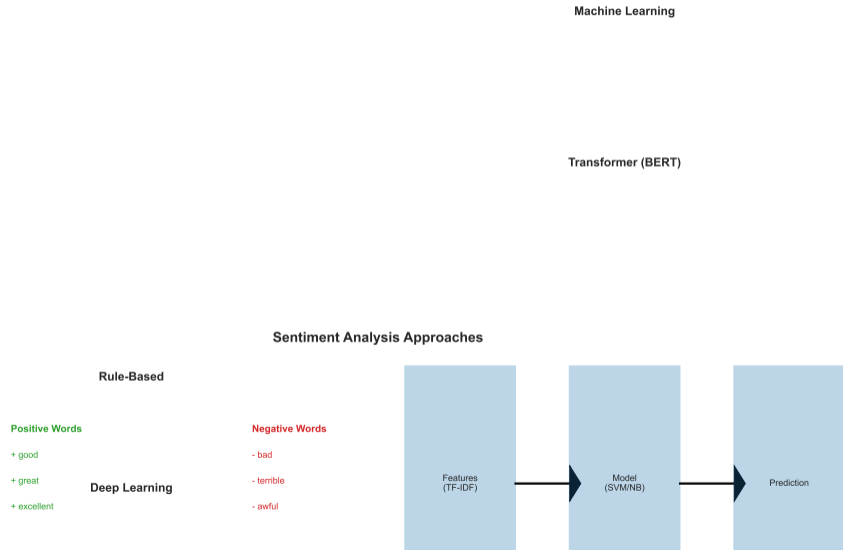
Fine-tuning (You):

- Your specific task
- Much smaller dataset

Different Sizes for Different Needs



From Rules to Deep Learning



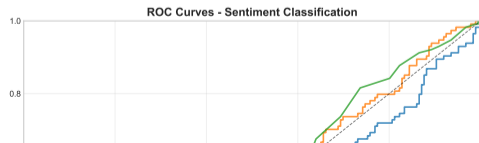
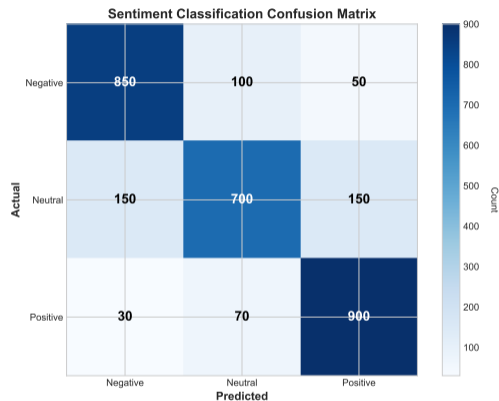
Choosing the Right Metrics

Classification Metrics:

- **Accuracy:** Overall correctness
- **Precision:** When you predict positive, how often correct?
- **Recall:** Of all positives, how many found?
- **F1-Score:** Harmonic mean of P&R
- **AUC-ROC:** Discrimination ability

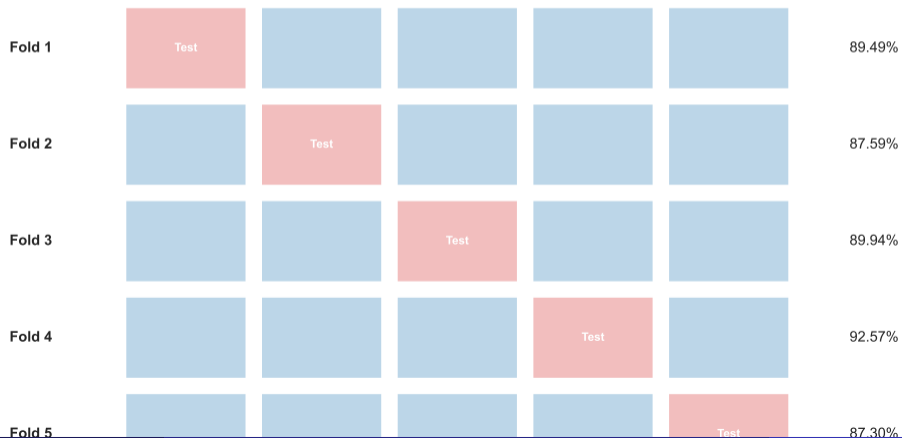
For Imbalanced Data:

- Weighted F1
- Macro/Micro averaging
- Cohen's Kappa

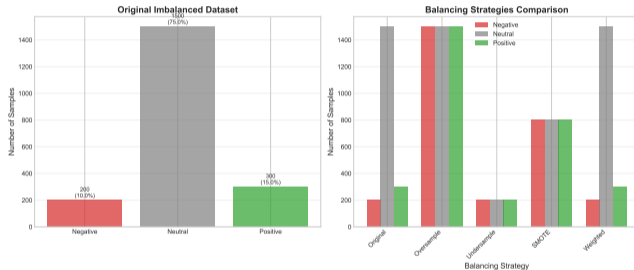


Avoiding Overfitting in NLP

5-Fold Cross-Validation for Text Data



When Negatives Dominate



Common Scenario:

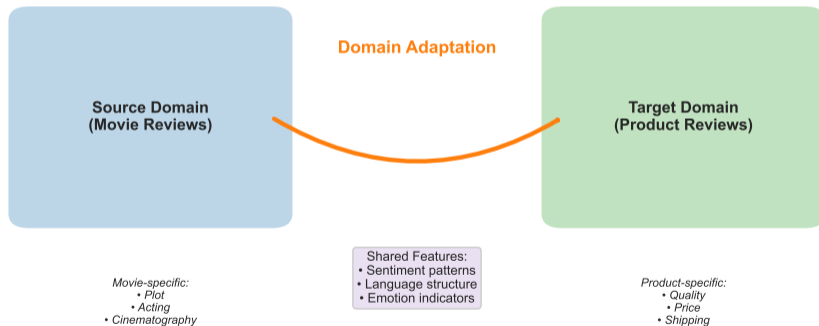
- 80% negative reviews
- 15% neutral
- 5% positive

Solutions:

- Class weights
- SMOTE oversampling
- Focal loss
- Ensemble methods
- Threshold tuning

Transfer Learning for NLP

Domain Adaptation in NLP

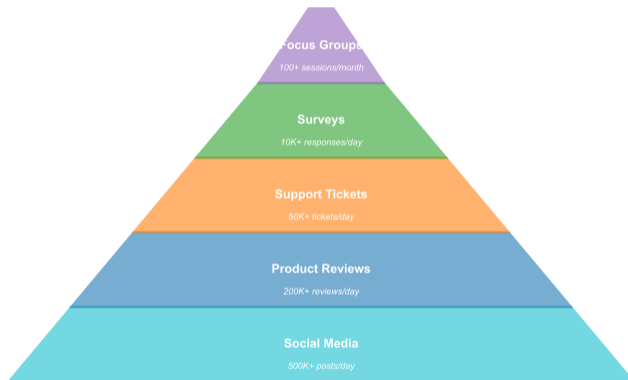


Part 3: Implementation

From Theory to Practice

Sources of User Language

Text Data Sources Hierarchy



Internal Sources:

- Customer reviews
- Support tickets
- Survey responses
- Chat transcripts
- Forum posts

External Sources:

- Social media
- App store reviews
- Reddit discussions
- News mentions

Handling Imperfect Data

Common Issues:

- HTML tags: `<p>text</p>`
- Emojis:
- URLs: `http://example.com`
- Mentions: `@user`
- Hashtags: `#awesome`
- Special chars: `…`

Python Cleaning:

```
1 import re
2 from bs4 import BeautifulSoup
3
4 def clean_text(text):
5     # Remove HTML
6     text = BeautifulSoup(text, "html.parser").text
7     # Remove URLs
8     text = re.sub(r'http\S+', '', text)
9     # Remove mentions
10    text = re.sub(r'@\w+', '', text)
11    # Normalize whitespace
12    text = ' '.join(text.split())
13    return text
```

Text Cleaning Pipeline

Raw Input



OMG!!! This product is AMAZINGGG ☐☐☐ #BestEver http://link.co

Lower & Strip



omg!!! this product is amazinggg ☐☐☐ #bestever

Creating Meaningful Features

Text Features:

- Length statistics
- Punctuation patterns
- Capitalization ratio
- N-grams (bigrams, trigrams)
- Part-of-speech tags
- Named entities

Sentiment Features:

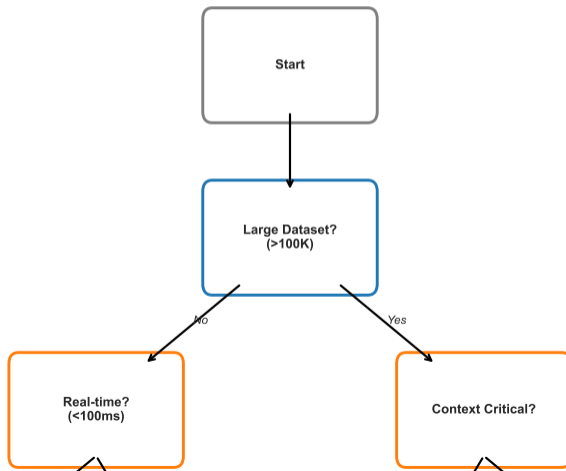
- Polarity scores
- Subjectivity measures
- Emotion intensities
- Negation handling

NLP Feature Engineering Hierarchy



Decision Framework

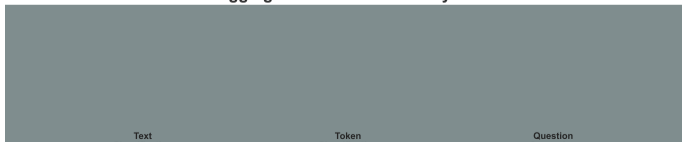
NLP Model Selection Flowchart



Modern NLP in 10 Lines

```
1 from transformers import pipeline
2
3 # Load pre-trained sentiment model
4 classifier = pipeline("sentiment-analysis")
5
6 # Single prediction
7 result = classifier("I love this product!")
8 # [{'label': 'POSITIVE', 'score': 0.9998}]
9
10 # Batch processing
11 texts = ["Great service!", "Terrible experience", "It's okay"]
12 results = classifier(texts)
13
14 # Custom model
15 classifier = pipeline("sentiment-analysis",
16                       model="nlp-town/bert-base-multilingual-uncased-sentiment")
```

HuggingFace Model Hub Ecosystem



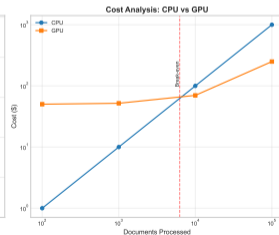
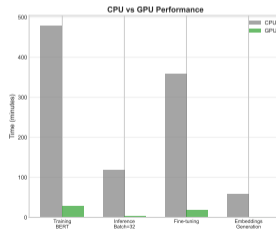
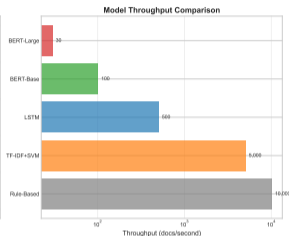
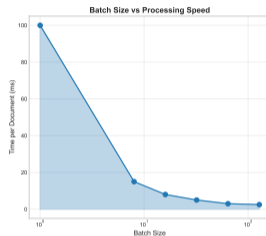
From Prototype to Production

Optimization Strategies:

- Batch inference
- GPU acceleration
- Model quantization
- Caching predictions
- Async processing
- Model distillation

Performance Gains:

- Single: 100 texts/min
- Batch: 5,000 texts/min
- GPU: 20,000 texts/min
- Distilled: 2x faster



Making Sense of Model Output

NLP Result Interpretation Framework

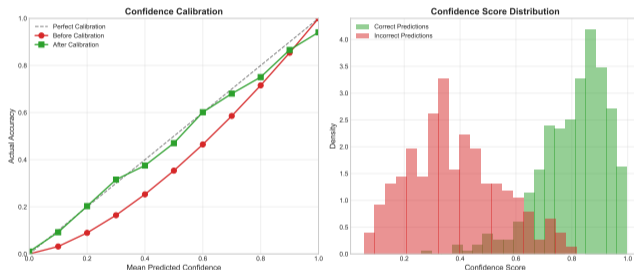
Input: "The product quality is excellent but the price is too high"



Interpretation:

Customer appreciates product quality but finds it overpriced.
Consider pricing strategy or highlight value proposition.

Uncertainty Quantification



Calibration Techniques:

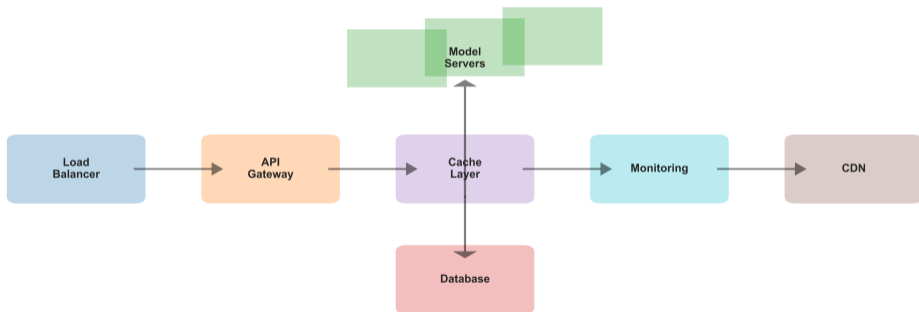
- Temperature scaling
- Platt scaling
- Isotonic regression
- Ensemble uncertainty

Decision Rules:

- High conf (≥ 0.9): Auto-process
- Medium (0.6-0.9): Flag for review
- Low (≤ 0.6): Human review

From Notebook to API

Production NLP Deployment Architecture



Making NLP Accessible

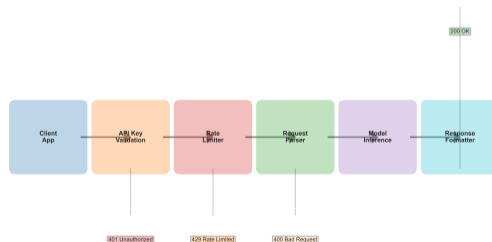
REST API Example:

```
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3
4 app = FastAPI()
5 model = load_model()
6
7 class TextRequest(BaseModel):
8     text: str
9
10 @app.post("/sentiment")
11 def analyze(request: TextRequest):
12     result = model(request.text)
13     return {
14         "sentiment": result["label"],
15         "confidence": result["score"]
16     }
```

Integration Points:

- CRM systems
- Analytics dashboards
- Support tickets
- Product reviews
- Social media feeds

NLP API Integration Flow

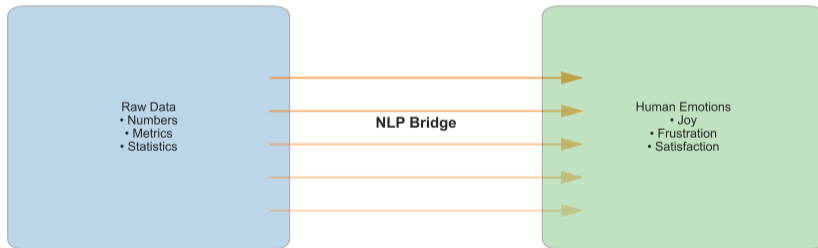


Part 4: Design Applications

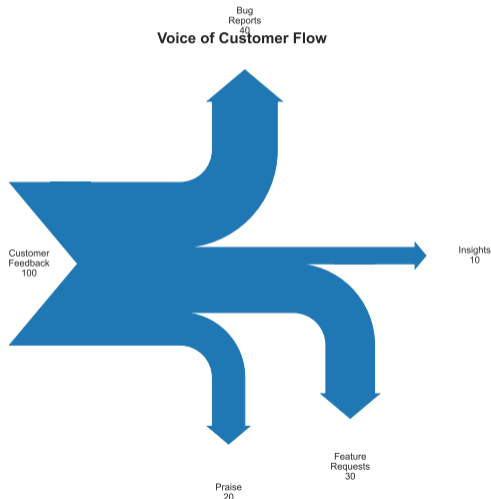
Transforming Emotions into Experiences

Making Numbers Human Again

Bridging Data and Emotions



What Users Really Say



VoC Framework:

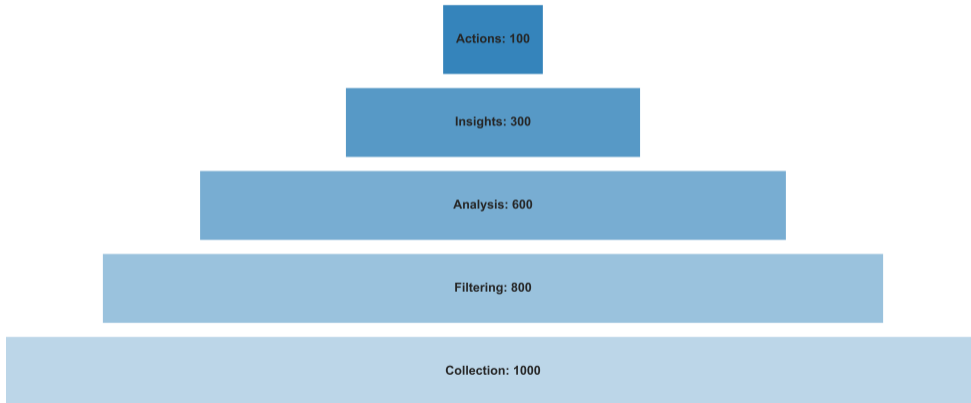
- Capture: All touchpoints
- Analyze: Sentiment & themes
- Categorize: By impact
- Prioritize: By frequency
- Act: Design changes

Key Metrics:

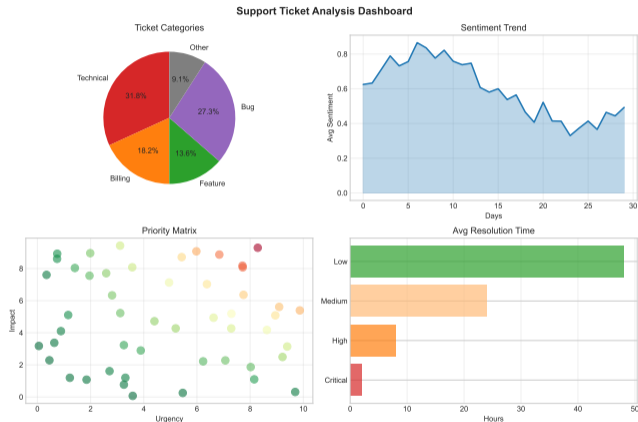
- Sentiment trends
- Topic frequency
- Emotion intensity
- Issue urgency

Systematic Insight Extraction

Review Mining Process Funnel



Understanding Frustration Patterns



Ticket Insights:

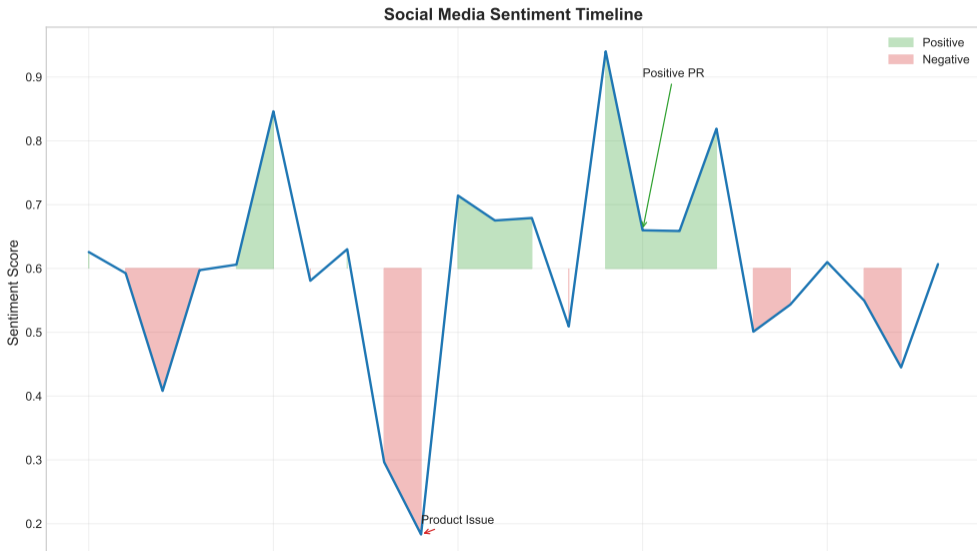
- Urgency detection
- Problem categorization
- Emotion escalation
- Resolution impact

Design Actions:

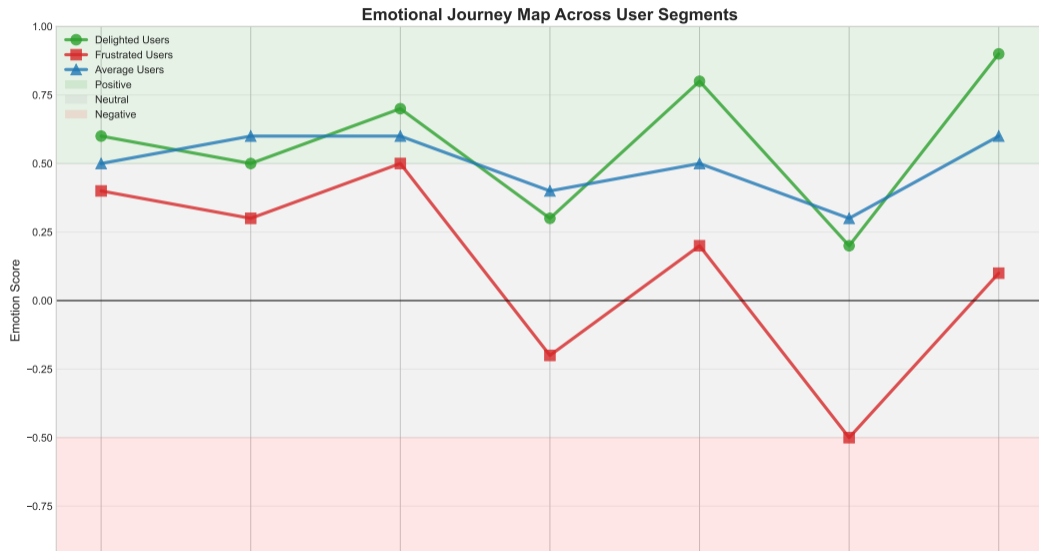
- Fix pain points
- Improve documentation
- Redesign workflows
- Prevent issues

Finding: 40% of angry tickets mention same navigation issue

Understanding Public Perception



Feelings Throughout the Experience



Systematic Problem Detection

Customer Pain Point Heatmap



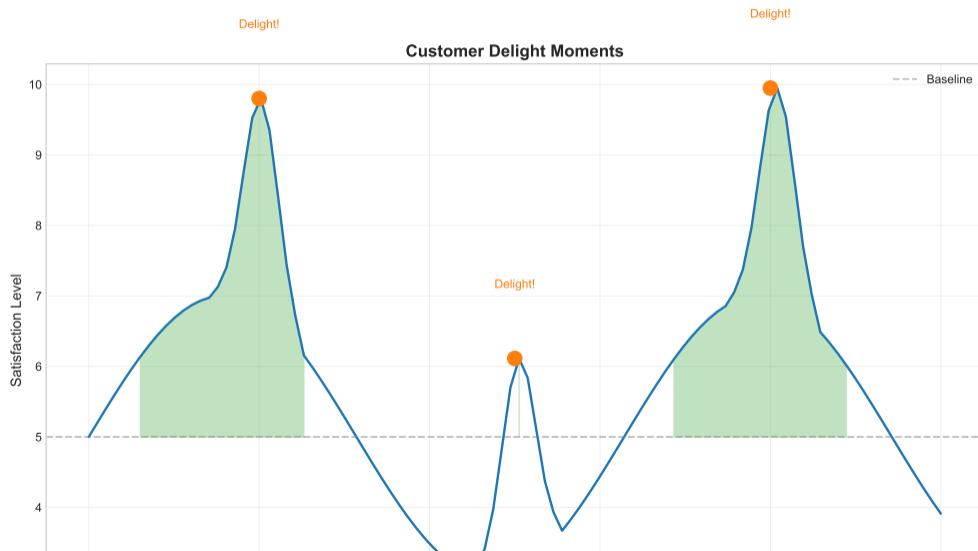
Pain Categories:

- Functional: Doesn't work
- Financial: Too expensive
- Process: Too complex
- Support: Can't get help
- Emotional: Feels bad

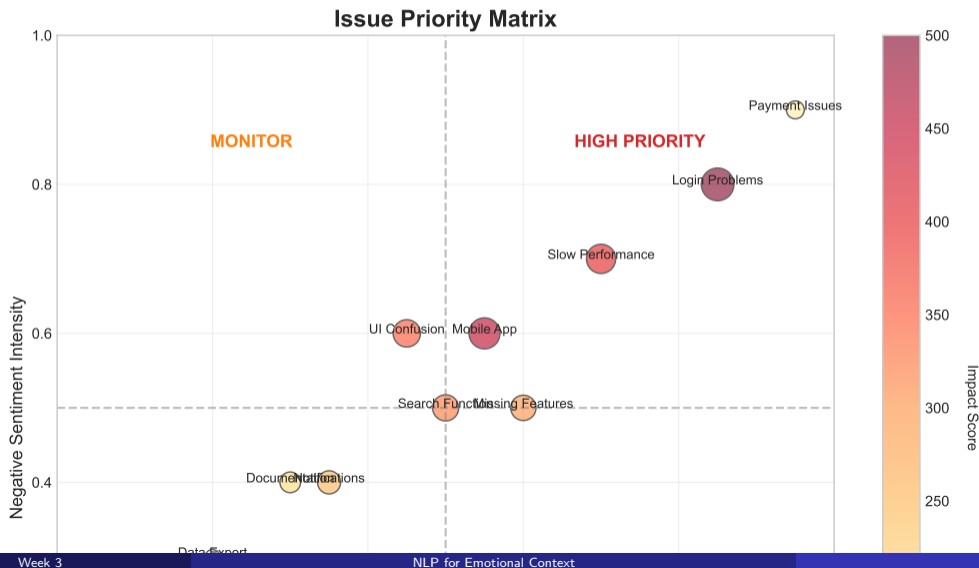
Severity Scoring:

- Frequency \times Impact
- Emotion intensity
- Business cost
- Churn correlation

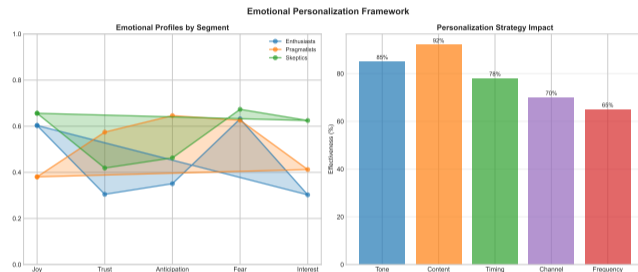
Amplifying the Positive



Data-Driven Design Decisions



Responding to User State



Adaptation Strategies:

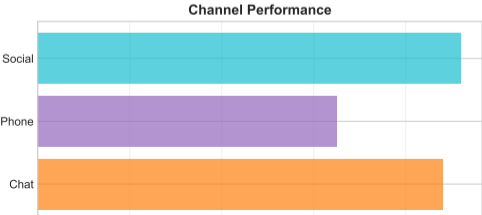
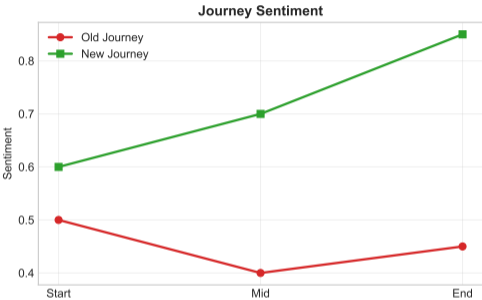
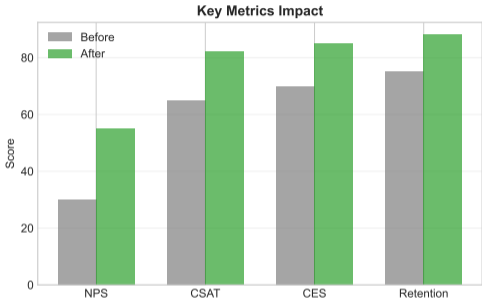
- Frustrated → Simplify
- Happy → Upsell
- Confused → Guide
- Angry → Escalate
- Satisfied → Engage

Implementation:

- Real-time detection
- UI adaptation
- Content variation
- Support routing

Tracking Design Success

NLP Impact Measurement Dashboard



ROI Summary

Investment: \$100,000
Revenue Gain: \$450,000
Cost Savings: \$200,000

Total ROI: 550%
Payback: 3 months

Part 5: Case Study & Practice

Amazon Review Intelligence System

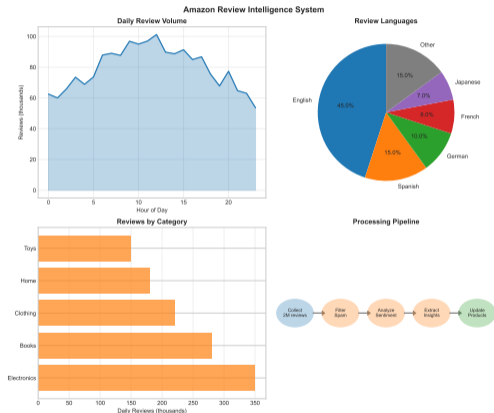
Understanding 100 Million Customers

The Challenge:

- 2+ million reviews daily
- 35 languages
- Multiple product categories
- Fake review detection
- Real-time insights needed

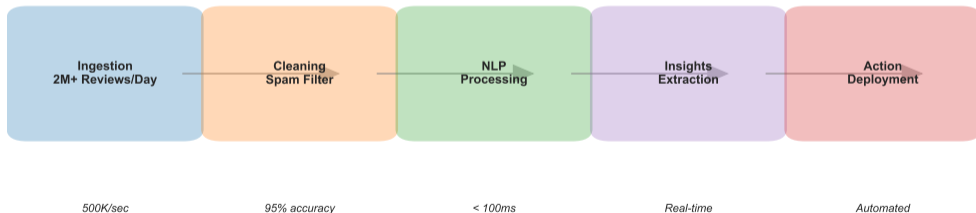
Business Goals:

- Improve product quality
- Identify trends early
- Enhance customer satisfaction
- Reduce return rates

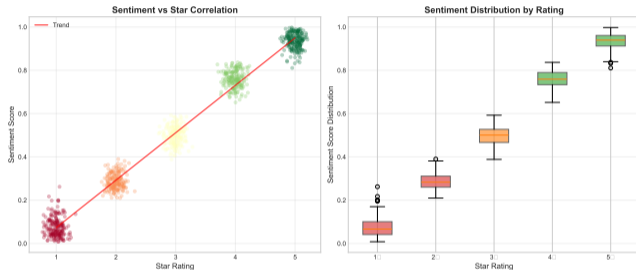


Processing Millions of Reviews

Amazon Review Processing Pipeline



Text Reveals Hidden Insights



Key Findings:

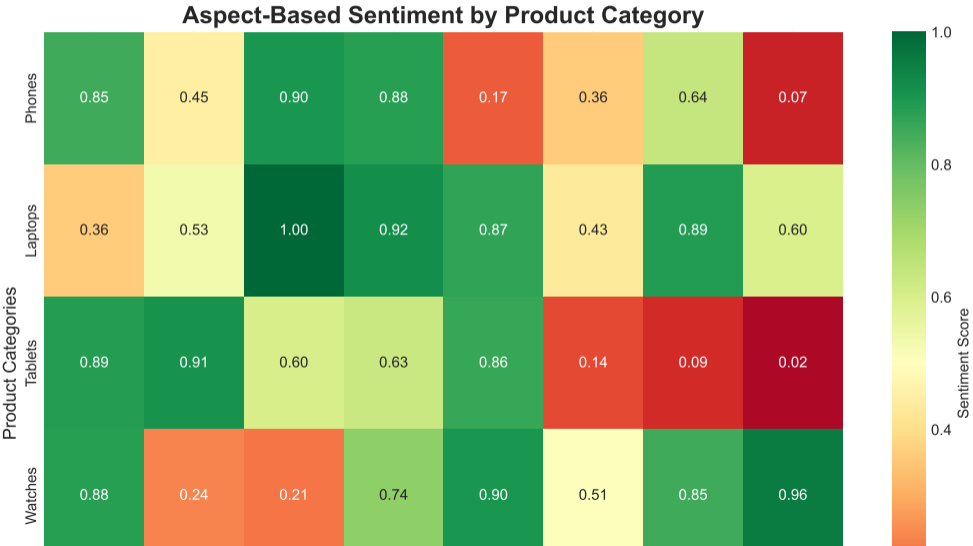
- 23% of 4-star reviews contain negative sentiment
- 5-star reviews with “but” = future problems
- 3-star most informative

Hidden Patterns:

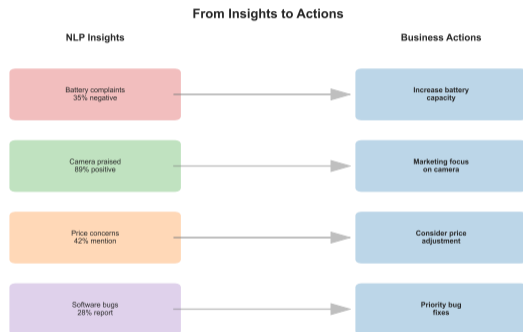
- Expectation mismatch
- Feature complaints
- Quality over time
- Comparison mentions

“5 stars! Amazing product but the app could be better” → Feature request identified

What Exactly Do They Love/Hate?



From Analysis to Action



Top Discoveries:

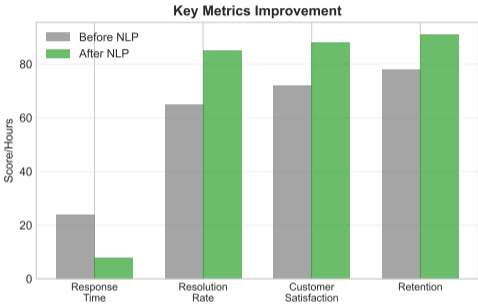
- 1 Setup wizard needed
- 2 Size expectations wrong
- 3 Packaging frustration
- 4 Feature discovery issue
- 5 Comparison shopping

Actions Taken:

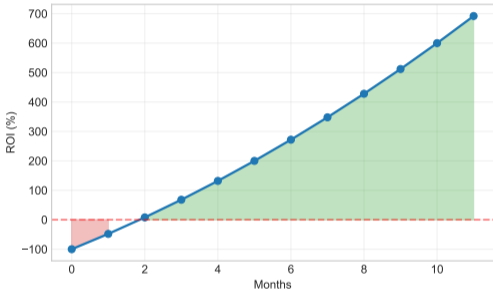
- Redesigned onboarding
- Added size reference
- Simplified packaging
- Created tutorials
- Comparison tool

ROI of Sentiment Analysis

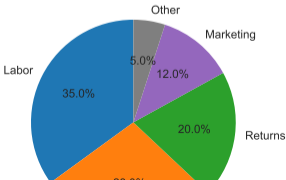
NLP Implementation Impact Metrics



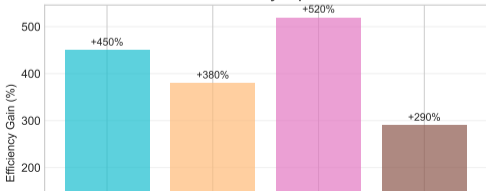
ROI Timeline



Cost Savings Breakdown



Process Efficiency Improvements



Your Turn: Analyze Brand Perception

Dataset:

- 5,000 tweets about a product
- Last 30 days
- Multiple languages
- Includes replies/mentions

Tasks:

- 1 Clean and preprocess
- 2 Sentiment classification
- 3 Emotion detection
- 4 Topic extraction
- 5 Temporal analysis
- 6 Create insights report

Deliverables:

- Jupyter notebook
- Sentiment dashboard
- Top 10 pain points
- Emotion journey map
- Recommendations

Bonus Challenges:

- Detect sarcasm
- Find influencers
- Predict virality
- Compare competitors

Time: 2 hours — **Tools:** Python, BERT, Plotly — **Goal:** Actionable insights

What You've Learned

Technical Skills:

- Text preprocessing pipelines
- BERT implementation
- Sentiment classification
- Emotion detection
- Production deployment

Tools Mastered:

- Hugging Face Transformers
- spaCy/NLTK
- FastAPI
- Pandas/NumPy

Design Applications:

- Voice of Customer analysis
- Emotional journey mapping
- Pain point prioritization
- Feature discovery
- Impact measurement

Business Value:

- Scalable insights
- Real-time monitoring
- Data-driven decisions
- Customer empathy

Next Week: Classification for Problem Definition

Measuring Word Importance

Term Frequency (TF):

$$TF(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

Inverse Document Frequency (IDF):

$$IDF(t, D) = \log \left(\frac{\text{Total number of documents } |D|}{\text{Number of documents containing term } t} \right)$$

TF-IDF Score:

$$TF-IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

High TF-IDF = Term is frequent in document but rare in corpus = Important/distinctive

Learning Word Representations

Skip-gram Objective:

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t; \theta)$$

where:

$$p(w_o | w_l) = \frac{\exp(v_{w_o}^T v_{w_l})}{\sum_{w=1}^W \exp(v_w^T v_{w_l})}$$

CBOW (Continuous Bag of Words):

$$p(w_t | w_{t-c}, \dots, w_{t+c}) = \frac{\exp(v_{w_t}^T \bar{v})}{\sum_{w=1}^W \exp(v_w^T \bar{v})}$$

where $\bar{v} = \frac{1}{2c} \sum_{-c \leq j \leq c, j \neq 0} v_{w_{t+j}}$ is the average context vector

Skip-gram: Predict context from word

CBOW: Predict word from context

The Core of Transformers

Attention Function:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

where:

- $Q \in \mathbb{R}^{n \times d_k}$: Query matrix
- $K \in \mathbb{R}^{m \times d_k}$: Key matrix
- $V \in \mathbb{R}^{m \times d_v}$: Value matrix
- d_k : Dimension of keys (scaling factor)

Multi-Head Attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

where:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Scaling by $\sqrt{d_k}$ prevents softmax saturation with large dimensions

Optimizing Sentiment Classification

Binary Cross-Entropy (Binary Sentiment):

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

Categorical Cross-Entropy (Multi-class):

$$\mathcal{L}_{CCE} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C y_{ij} \log(\hat{y}_{ij})$$

Focal Loss (Imbalanced Data):

$$\mathcal{L}_{FL} = -\alpha_t (1 - p_t)^\gamma \log(p_t)$$

where:

- α_t : Class weight
- γ : Focusing parameter (typically 2)
- p_t : Model's estimated probability for correct class

F1 Score (Evaluation):

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$