Week 2: What are users really saying?

Understanding emotions in text with NLP

ML/AI/GenAI for Design Thinking

BSc Course - 12 Week Program

2024

The Challenge: Text Has Hidden Emotions

What users write:

- "Great product... if you like disappointment"
- "Absolutely perfect! Never worked once"
- "Can't complain" (literally can't)
- "Fine." (but are they really?)

What they actually mean:

- Frustrated and sarcastic
- Extremely angry
- Forced acceptance
- Deeply unsatisfied

Human language is complex: sarcasm, context, subtle emotions

Traditional Approach: Keyword Matching

Rule-based sentiment:

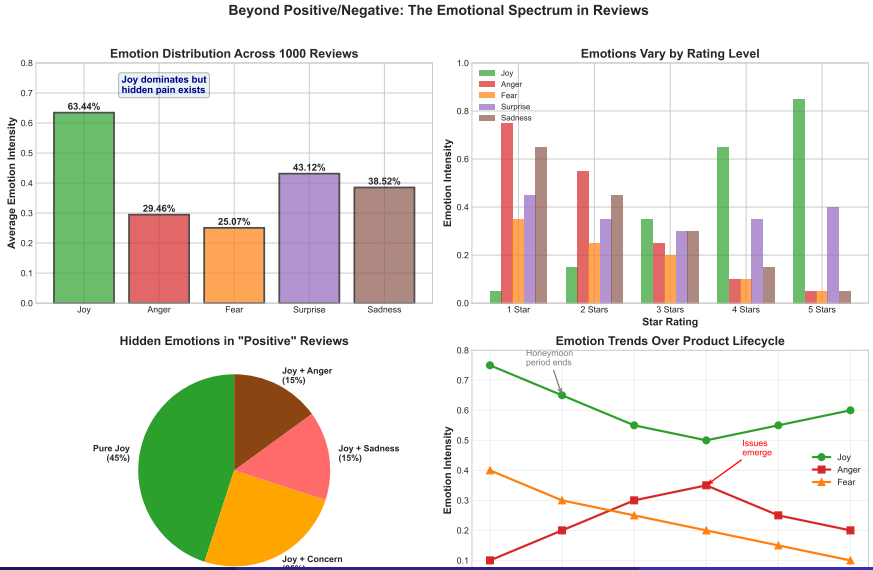
- Count positive words (+1)
- Count negative words (-1)
- Sum the scores
- Classify as pos/neg/neutral

Why it fails:

- "Not bad" → Negative (wrong!)
- "Terribly good" → Mixed (wrong!)
- Misses context completely
- Can't detect sarcasm

Example failure:
"I love waiting 3 hours for support. Really enhances the experience."
Rule-based: POSITIVE — Reality: NEGATIVE (sarcastic)

What We're Missing: The Emotional Spectrum



Real Case: Amazon Review Analysis

One product, 5,000 reviews, vastly different emotions:

5-star review:
"Disappointed it took so long to find this! Game changer!"
Traditional: Mixed
BERT: Joy + Regret

3-star review:
"Works as advertised. Nothing special."
Traditional: Neutral
BERT: Mild disappointment

1-star review:
"Perfect! If you enjoy throwing money away."
Traditional: Positive
BERT: Sarcasm, Anger

The Question: How can we understand these emotions at scale?

Why This Matters: The Cost of Misunderstanding

Business Impact:

- 68% of customers leave due to perceived indifference
- Missing early warning signs
- Wrong product decisions
- Failed innovations

Real Example: Microsoft Clippy

- Assumed users wanted help (wrong emotion)
- Actually caused frustration and annoyance
- Failed to read emotional context
- Result: Most hated feature in software history

Design Impact:

- Building for wrong emotions
- Missing user pain points
- Solving non-problems
- Creating frustrating experiences

Natural Language Processing: Teaching Machines to Read

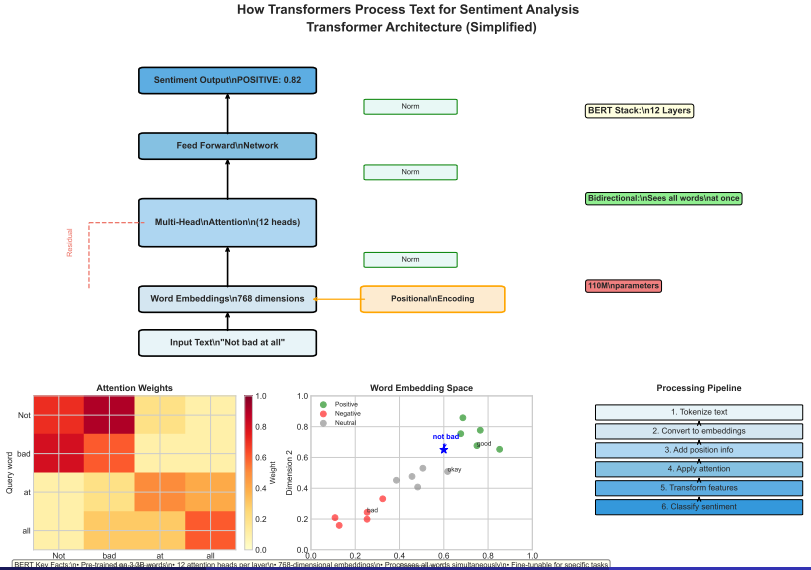
The Evolution of Understanding:

1. **1950s - Rules:** "If word = 'good' then positive"
2. **1990s - Statistics:** Count word frequencies
3. **2013 - Word Vectors:** Words as numbers (Word2Vec)
4. **2018 - Transformers:** Understanding context (BERT)
5. **2023 - LLMs:** Full language understanding (GPT-4)

Old way: Words in isolation
"bank" = financial institution

Transformer way: Words in context
"river bank" vs "investment bank"

The Transformer Revolution (2017)



BERT: Bidirectional Understanding

BERT = Bidirectional Encoder Representations from Transformers

Traditional (Left-to-right):

- Reads: "The → movie → was → "
- Predicts next word
- Misses future context

BERT (Bidirectional):

- Sees entire sentence at once
- "The movie was [MASK] boring"
- Uses both past and future context

Pre-trained on:

- 3.3 billion words
- Wikipedia + BookCorpus
- Learned general language understanding
- Can be fine-tuned for specific tasks

How BERT Analyzes Sentiment

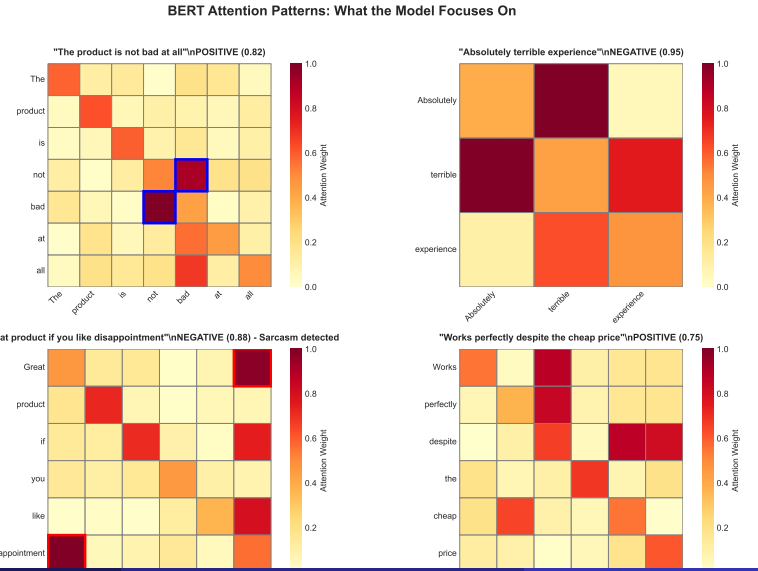
Example: "The product is not bad at all"

1. **Tokenization:** [The] [product] [is] [not] [bad] [at] [all]
2. **Embedding:** Each word $\rightarrow$ 768-dimensional vector
3. **Attention:** Focus on "not" + "bad" combination
4. **Context:** Understand double negative = positive
5. **Output:** Sentiment score: 0.82 (Positive)

What makes BERT special:

- Understands negation ("not bad" = good)
- Catches intensifiers ("very", "extremely")
- Recognizes sarcasm patterns
- Considers word order and context

Attention: What BERT Focuses On



Live Demo: Analyzing Real Reviews

```
from transformers import pipeline

# Load pre-trained sentiment model
analyzer = pipeline("sentiment-analysis",
                    model="bert-base-uncased")

# Real product reviews
reviews = [
    "Great product... if you like disappointment",
    "Absolutely worth every penny!",
    "Not the worst thing I've bought",
    "Fine, I guess..."
]

# Analyze each review
for review in reviews:
    result = analyzer(review)
    print(f"{review[:30]}... -> {result}")
```

Output: Real sentiment scores with confidence levels

Multi-dimensional Sentiment Analysis

BERT can detect multiple emotions simultaneously:

Review Text	Joy	Anger	Fear	Surprise	Sadness
"Amazing! Exceeded expectations!"	0.92	0.01	0.02	0.65	0.01
"Worried about the warranty"	0.05	0.02	0.78	0.10	0.15
"Broke after 2 days. Unbelievable."	0.01	0.85	0.05	0.72	0.45
"Works okay, nothing special"	0.15	0.05	0.02	0.01	0.35

Insights from emotion detection:

- Identify specific pain points (fear about warranty)
- Detect delight factors (surprise + joy)
- Find frustration patterns (anger + sadness)
- Measure emotional intensity

Domain Adaptation: Making BERT Yours

Generic BERT → Your Industry BERT

Pre-trained BERT:

- General language understanding
- Common sentiment patterns
- Wikipedia knowledge

Fine-tuned BERT:

- Your industry jargon
- Product-specific sentiment
- Customer language patterns

Example: Gaming Industry

- "Lag" → Negative (technical issue)
- "Grind" → Context-dependent (positive for RPG fans)
- "OP" → Positive (overpowered = fun)
- "Nerf" → Negative (reduced power)

Fine-tuning process: 1000 labeled examples → 2 hours → Custom model

BERT vs Traditional: The Numbers

BERT's Limitations: What It Can't Do

Still struggles with:

- **Cultural context:** Idioms, slang, regional expressions
- **Emerging language:** New memes, Gen-Z speak
- **Multi-lingual:** Code-switching, mixed languages
- **Visual context:** Emojis, images, formatting
- **Domain expertise:** Deep technical discussions

Computational requirements:

- Model size: 440MB (BERT-base)
- Processing: 100 reviews/second on GPU
- Fine-tuning: 2-4 hours on cloud

Remember: BERT is a tool, not a replacement for human understanding

Empathize - Feel: Beyond Who to How

Traditional Empathy:

- Who are our users?
- What do they need?
- Demographics and behaviors
- Surface-level understanding

Emotional Empathy:

- How do users feel?
- Why do they feel that way?
- Emotional states and triggers
- Deep psychological understanding

Example: Food Delivery App

- Traditional: "Users aged 25-35 order dinner"
- Emotional: "Exhausted parents feeling guilty about not cooking"
- Design implication: Add "healthy kids meals" and remove guilt triggers

The Digital Empathy Gap

Why digital products often feel cold:

1. **Scale challenge:** Can't interview millions
2. **Distance:** Never meet users face-to-face
3. **Assumptions:** Think we know how users feel
4. **Metrics focus:** Clicks over emotions

The result:

- Frustrating error messages
- Tone-deaf notifications
- Insensitive default options
- Missing emotional moments
- Generic experiences
- High abandonment rates
- Poor reviews
- Lost loyalty

NLP bridges this gap: Understand feelings from text at scale

Every Review is a Conversation

Users are telling us how they feel - we just need to listen

What they write	What they're really saying
"Loading takes forever"	I'm frustrated and impatient
"Finally, someone gets it!"	I feel understood and validated
"Why is this so complicated?"	I'm confused and need help
"This made my day"	I'm delighted and grateful
"I give up"	I'm exhausted and defeated

Design responses to emotions:

- Frustration → Simplify and speed up
- Confusion → Better onboarding and tooltips
- Delight → Amplify these moments
- Defeat → Provide encouragement and support

Mapping the Emotional Journey

Traditional Journey Map + Emotional Data = Insight Gold

Stage	Action	Traditional Metric	Emotional State
Discover	Find product	Page views	Curious (65%), Skeptical (35%)
Evaluate	Read reviews	Time on page	Anxious (40%), Hopeful (60%)
Purchase	Checkout	Conversion	Excited (70%), Worried (30%)
Receive	Unbox	-	Delighted (80%), Disappointed (20%)
Use	First week	Retention	Frustrated (45%), Satisfied (55%)

Insights from emotional journey:

- High anxiety at evaluation → Add trust signals
- Frustration in first week → Improve onboarding
- Delight at unboxing → Invest in packaging

Translating ML Output to Design Insight

The Translation Process:

1. **Collect:** Gather text from all touchpoints
2. **Analyze:** Run BERT sentiment analysis
3. **Cluster:** Group similar emotions
4. **Interpret:** Understand root causes
5. **Design:** Create emotional solutions

Example: E-learning Platform

- Data: 40% frustration in week 2
- Root cause: Difficulty spike too steep
- Design solution: Add intermediate lessons
- Result: Frustration down to 15%, completion up 25%

Case Study: Airbnb's Emotional Design

How Airbnb used sentiment analysis to redesign reviews

The Problem:

- Hosts felt attacked by negative reviews
- Guests felt unheard with issues
- Trust eroding on both sides

The Analysis:

- Analyzed 50M reviews with NLP
- Found emotion patterns: Fear, anger, disappointment
- Identified trigger words and phrases

The Solution:

- Added emotion detection to review system
- Prompted constructive language for negative emotions
- Created "private feedback" for sensitive issues
- Result: 23% reduction in hostile reviews, 15% increase in host satisfaction

Your Emotional Analysis Toolkit

Data Sources:

- Reviews and ratings
- Support tickets
- Social media mentions
- Survey responses
- Chat transcripts

Analysis Tools:

- Hugging Face Transformers (free, powerful)
- Google Cloud Natural Language API
- AWS Comprehend
- Custom fine-tuned models

Visualization:

- Emotion heat maps
- Sentiment timelines
- Word clouds with feeling colors
- Journey emotion graphs

The Perfect Partnership: Human + AI

What AI Does Well:

- Process thousands of texts
- Find patterns consistently
- Work 24/7 without fatigue
- Quantify emotions
- Track changes over time

What Humans Do Well:

- Understand context deeply
- Feel genuine empathy
- Make creative connections
- Design solutions
- Validate insights

Together: Scale + Depth = True Digital Empathy

Best practice: AI finds patterns → Humans investigate why → Together design solutions

Let's Build: Sentiment Analysis Pipeline

5-minute implementation:

```
# Step 1: Install and import
!pip install transformers
from transformers import pipeline

# Step 2: Load model
sentiment = pipeline("sentiment-analysis")

# Step 3: Analyze your data
reviews = ["Your actual product reviews here"]
results = sentiment(reviews)

# Step 4: Extract insights
emotions = {"positive": 0, "negative": 0}
for r in results:
    emotions[r['label'].lower()] += 1

print(f"Customer sentiment: {emotions}")
```

Your turn: Try with your product's reviews!

Finding Emotional Patterns in Your Data

What to look for:

1. **Emotional clusters:** Groups with similar feelings
2. **Trigger events:** What causes emotional shifts
3. **Sentiment trends:** How feelings change over time
4. **Outliers:** Extremely positive/negative responses

Pattern → Insight → Action:

- Pattern: Anger spikes on Mondays
- Insight: Weekend issues not resolved
- Action: Add weekend support or Monday priority queue

- Pattern: Joy in unboxing videos
- Insight: Physical experience matters
- Action: Invest in packaging design

From Sentiment to Empathy Maps

AI-Powered Empathy Map:

Think & Feel
(from sentiment)
- Frustrated 45%
- Hopeful 30%
- Anxious 25%

Hear
(from mentions)
- "Too complex"
- "Worth the price"
- "Needs tutorial"

Say & Do
(from reviews)
- Complain about speed
- Praise design
- Request features

Pain/Gain
(from analysis)
Pain: Learning curve
Gain: Time saved

Traditional: 20 interviews $\rightarrow$ 1 empathy map
With NLP: 10,000 reviews $\rightarrow$ 5 segment empathy maps $\rightarrow$ Same day

What We Achieved Today

We learned to hear emotions at scale:

- Understood how BERT reads sentiment in context
- Moved beyond positive/negative to emotional spectrum
- Detected sarcasm and complex feelings
- Connected emotions to design decisions
- Built a sentiment analysis pipeline

The transformation:

Before:

- Reading 100 reviews manually
- Missing emotional nuance
- Guessing how users feel

After:

- Analyzing 10,000 reviews automatically
- Understanding complex emotions
- Knowing exactly how users feel

Result: Deep emotional understanding at scale

Next Week's Challenge

We understand emotions in individual reviews...

But what about the patterns across thousands?

The new questions:

- Which words trigger strong emotions?
- What phrases appear in happy vs angry reviews?
- How do we find the "moments that matter"?
- Can we identify the root causes automatically?

Next Week: Attention Mechanisms - Finding What Matters Most
How AI learns to focus on the important parts and ignore the noise

Resources & Your Assignment

This Week's Challenge:

1. Find 100 reviews of a product you care about
2. Run sentiment analysis using the code provided
3. Identify 3 emotional patterns
4. Create one design recommendation based on emotions
5. Share your findings next week

Resources:

- Hugging Face Tutorial: huggingface.co/tasks/sentiment-analysis
- BERT Paper: arxiv.org/abs/1810.04805
- Google Colab notebook: [provided in class]

Remember:

- Every review is a user sharing their feelings
- NLP helps us listen at scale
- Emotions drive design decisions
- You now have superpowers - use them wisely!