Lesson 40: Sentiment Analysis

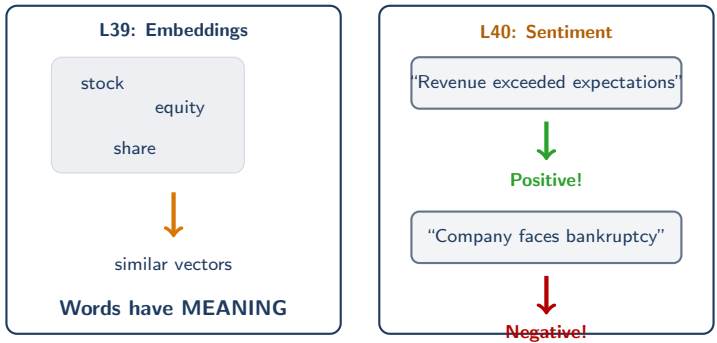
Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

Previously: Words That Know Their Neighbors



L39 gave us meaning. Now: what **opinion** does the text express?

From understanding words to understanding sentiment – the final NLP step

Learning Objectives

The Problem: Financial markets are driven by sentiment – fear, greed, optimism. How do we automatically measure sentiment from text and use it as a trading signal?

After this lesson, you will be able to:

1. **Explain** lexicon-based vs ML-based sentiment approaches (Remember)
2. **Apply** VADER for quick sentiment scoring of social media (Apply)
3. **Compare** VADER vs FinBERT on financial text (Analyze)
4. **Design** a sentiment-based trading signal pipeline (Create)

Finance Application: News-based trading signals, earnings call analysis, social media monitoring

When Elon Tweets, Stocks Move

Real examples:

- 2021: "Tesla stock price is too high imo" $\rightarrow$ $-$**10%** same day
- 2022: Musk tweets dogecoin meme $\rightarrow$ DOGE **+25%** in hours
- 2023: "AI will change everything" $\rightarrow$ AI stocks rally

The opportunity:

- Millions of tweets, news articles, earnings calls every day
- No human can read them all – but an algorithm can
- Sentiment analysis = automating "reading the room" at scale

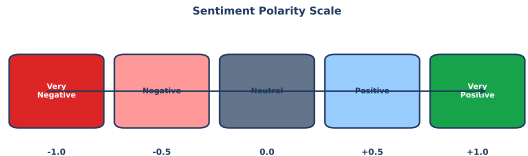
The question: Can we turn text mood into a trading signal?

Social media moves markets – sentiment analysis automates detection

What IS Sentiment Analysis?

The Simple Idea:

- Input: text (headline, tweet, earnings call transcript)
- Output: sentiment label (positive / negative / neutral) or score (-1 to $+1$)
- Quantifies "market mood" from unstructured text

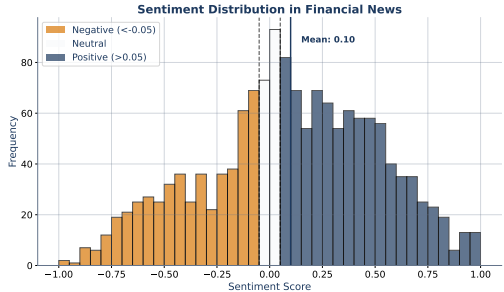


Sentiment analysis: converting text mood into numbers for quantitative models

Sentiment Score Distribution

Financial News Sentiment Patterns

- Financial news tends slightly positive (media bias)
- Neutral class is largest – most news is factual reporting
- Threshold at ± 0.05 for neutral classification



Knowing the base rate matters: most news is neutral, not extreme

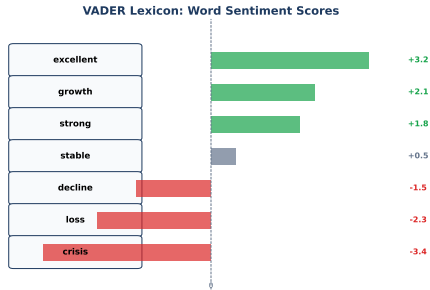
Three Approaches to Sentiment

1. **Lexicon-based (VADER):**
   - Look up words in sentiment dictionary (7,500+ scored words)
   - Fast, no training, good for social media
2. **ML-based (Naive Bayes, SVM, Logistic Regression):**
   - Train classifier on labeled data (TF-IDF features)
   - Better accuracy, requires labeled training data
3. **Transformer-based (FinBERT):**
   - Pre-trained deep learning model fine-tuned on financial text
   - Best accuracy for finance, but slowest

Trade-off: speed vs accuracy. VADER for volume, FinBERT for precision.

VADER: Rule-Based Sentiment

Dictionary: 7,500+ words scored ("great" = +3.1, "terrible" = −2.5)

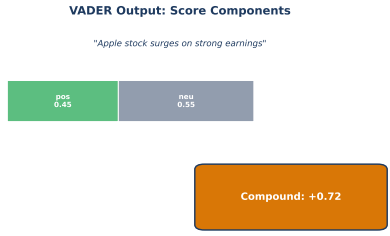


Smart rules: Negation ("not good" = negative), Intensifiers ("VERY good" > "good"), Punctuation ("Good!!!" > "Good.")

VADER: fast, no training needed. Use `compound` score for overall sentiment.

VADER: Scores in Practice

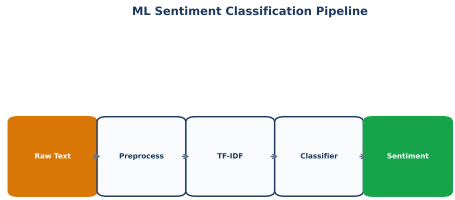
```
sia = SentimentIntensityAnalyzer()
sia.polarity_scores('Stock surged on strong earnings!')
→ {neg:  0.0, neu:  0.35, pos:  0.65, compound:  0.78}
```



Compound score: -1 (most negative) to $+1$ (most positive). Key metric.

ML Approach: TF-IDF + Classifier

Pipeline: Label data → TF-IDF features (L38) → Train/test split → Logistic Regression / Naive Bayes → Evaluate (accuracy, F1)



ML approach needs labeled data but often outperforms lexicon methods

What Words Drive Sentiment Predictions?

Top predictive features from TF-IDF + Logistic Regression:

- Positive: “growth”, “exceeded”, “strong”, “profit”
- Negative: “loss”, “decline”, “risk”, “downturn”
- Model learns domain-specific sentiment vocabulary

Inspecting feature weights reveals what the model considers positive vs negative

Checkpoint: Lexicon or ML?

Q1: You have 1 million tweets to score in real-time. VADER or ML model?

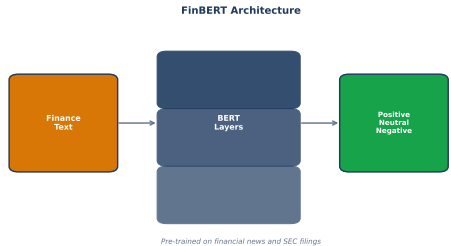
Q2: "Revenue declined less than expected." How would VADER score this?

Q3: What labeled data would you need for an ML sentiment classifier?

Answers: Q1: VADER (speed). Q2: Negative ("declined"), but it's actually positive context! Q3: Text + sentiment labels (pos/neg/neutral).

FinBERT: Finance-Specific Sentiment

Why general tools fail: VADER scores "revenue declined less than expected" as negative. FinBERT: **positive** (it understands "less than expected" = good).



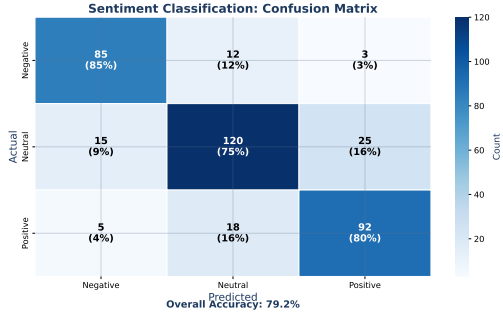

```
from transformers import pipeline
nlp = pipeline('sentiment-analysis', model='ProsusAI/finbert')
```

FinBERT: slower but much more accurate for financial text than VADER

Classification Performance

How Well Can We Classify Sentiment?

- Confusion matrix shows prediction accuracy per class
- Neutral class is hardest (ambiguous language)
- Positive and negative are more distinct

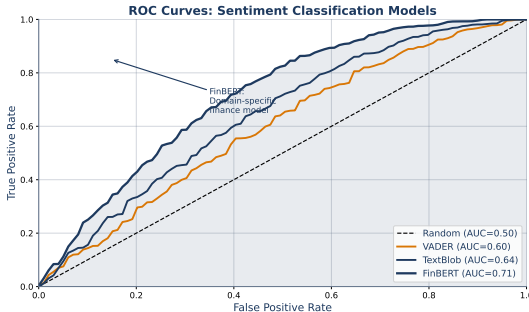


Overall accuracy: ∼79% – neutral class is the main challenge

Model Comparison: ROC Curves

Which Sentiment Model Performs Best?

- FinBERT outperforms general-purpose tools on financial text
- VADER is faster but less accurate for domain-specific language
- AUC summarizes overall model quality

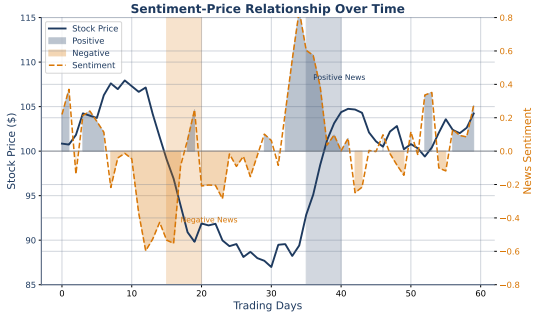


FinBERT AUC > ML classifier > VADER on financial text

Does Sentiment Predict Returns?

Sentiment-Price Relationship:

- News sentiment often leads price movements by hours or days
- Aggregate sentiment = market-wide mood indicator
- Individual stock sentiment = stock-specific signal

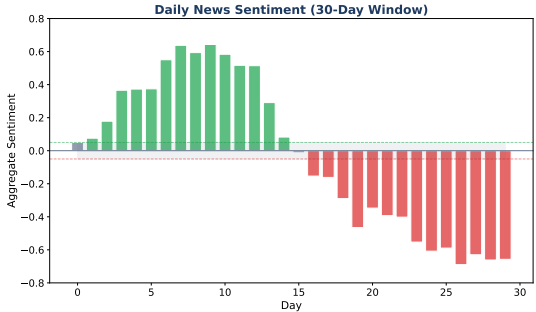


News sentiment often leads price movements – but the edge decays quickly

Finance: Headline Sentiment Scoring

Applying VADER to real financial headlines:

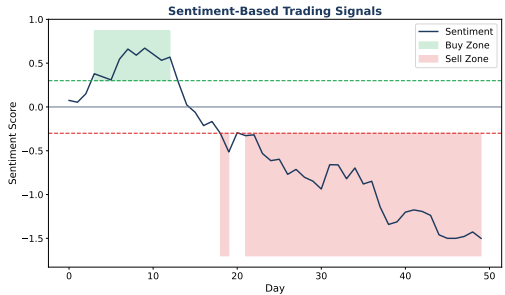
- Score each headline with compound score (-1 to $+1$)
- Aggregate daily scores per stock or market
- Track sentiment trends alongside price



Daily sentiment aggregation smooths noise from individual headlines

Building a Sentiment Trading Signal

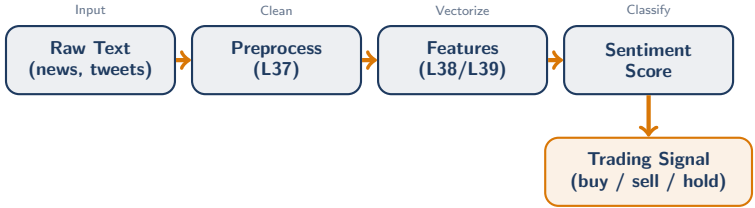
Pipeline: Collect news → Score (VADER/FinBERT) → Aggregate daily → Signal (high → long, low → short) → Backtest



Caveats: Signal decays in hours (not weeks). Backtesting bias. Transaction costs.

Sentiment signals are real but small – combine with other factors

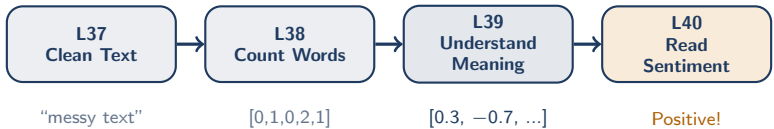
The Complete Sentiment Pipeline



Caveats: Signal decays in hours. Backtesting bias (real-time availability). Transaction costs erode small edges.

The full NLP pipeline: L37 clean → L38 vectorize → L39 embed → L40 classify

Module 8: The Journey from Text to Insight



Machines can read!

From raw text to actionable market signals in 4 lessons

Module 8 complete: text preprocessing → features → meaning → sentiment

Hands-On Exercise (25 min)

Task: Build a News Sentiment Analyzer

1. Apply VADER to 50 financial news headlines
2. Compare VADER compound scores with manual labels
3. Try FinBERT on the same headlines (if GPU available)
4. Plot sentiment score distribution
5. Create a simple buy/sell signal based on sentiment threshold

Deliverable: Sentiment distribution plot + accuracy comparison VADER vs FinBERT.

Extension: Correlate daily sentiment with next-day stock returns

Reading Between the Lines

What we've accomplished in Module 8:

- **L37:** Cleaned raw text into usable tokens
- **L38:** Converted tokens into numerical features (BOW, TF-IDF)
- **L39:** Gave words semantic meaning via embeddings
- **L40:** Extracted opinions and built trading signals

The big picture:

- Text data is the largest untapped data source in finance
- NLP turns unstructured text into structured features
- Sentiment is just one application – also: topic modeling, NER, summarization

---

NLP is a superpower: read thousands of documents in seconds

Key Takeaways

What you should remember:

1. **VADER:** fast, rule-based, good for social media (compound score)
2. **FinBERT:** slower but much more accurate for financial text
3. **ML pipeline:** TF-IDF + classifier needs labeled data but is flexible
4. **Sentiment signals** are real but decay fast – combine with other factors

Tool selection guide:

- High volume, low stakes $\rightarrow$ VADER
- Financial precision needed $\rightarrow$ FinBERT
- Custom domain, labeled data available $\rightarrow$ ML pipeline

---

Memory: VADER = fast/rule-based. FinBERT = accurate/finance-specific. Compound score.

Preview: From Models to Products

Module 8 is complete. You can now process and analyze text.

Next: How do we **deploy** our models so others can use them?

Module 9 – Deployment:

- **L41:** Model Serialization – saving and loading trained models
- **L42:** FastAPI – building REST APIs for ML models
- **L43:** Streamlit – interactive dashboards
- **L44:** Cloud Deployment – putting models in production

The course arc:

Data (M1–2) $\rightarrow$ Stats (M3) $\rightarrow$ ML (M4–6) $\rightarrow$ DL (M7) $\rightarrow$ NLP (M8) $\rightarrow$ **Deploy (M9)**

Next lesson: L41 Model Serialization – pickle, joblib, and model versioning