Quantifying Narratives and their Impact on Financial Markets

Complete Pipeline from News to Trading Signals

Based on Bhargava, Lou, Ozik, Sadka, Whitmore (2022)

State Street Associates & MKT MediaStats

January 2025

Presentation Overview

The Power of Narratives in Financial Markets

Robert Shiller's Narrative Economics

- "Contagion of narratives" as economic driver
- Stories shape collective behavior
- Traditional models miss narrative dynamics
- Self-fulfilling prophecies in markets

Research Questions

1. Can narratives be quantified systematically?
2. Do narratives explain market returns?
3. Can narratives predict future movements?
4. How to construct narrative portfolios?

This Research Contribution

- **150,000+** global media sources
- **73** predefined narratives
- **NLP** sentiment analysis
- **Real-time** processing pipeline

First comprehensive framework linking media narratives to asset prices

Historical Context: Evolution of Narrative Economics

Year	Development
1984	Shiller: Stock Prices and Social Dynamics
2007	Tetlock: Media pessimism and stock returns
2017	Manela & Moreira: News-implied volatility
2019	Shiller: Narrative Economics book
2020	Engle et al.: Climate change news hedging
2021	Mai & Pukthuanthong: 150 years NYT analysis
2022	**This work: Comprehensive narrative framework**
2024	BERTopic for financial narratives
2025	**LLMs with RAG for real-time processing**

Evolution from simple word counts to sophisticated NLP frameworks

Narrative Contagion Model

SIR Model for Narrative Spread

Let $S(t)$, $I(t)$, $R(t)$ denote susceptible, infected, and recovered populations:

$$\frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t)$$

where:

- β = transmission rate
- γ = recovery rate
- $R_0 = \beta/\gamma$ = basic reproduction number

Market Impact

$$r_{i,t} = \alpha + \sum_n \beta_n \cdot NI_{n,t} + \epsilon_{i,t}$$

Behavioral Mechanisms

- **Availability Heuristic**: Recent narratives overweighted
- **Confirmation Bias**: Selective narrative attention
- **Herding**: Social proof amplification
- **Representativeness**: Pattern over-extrapolation

Empirical Predictions

1. Narrative intensity ⇒ volatility
2. Sentiment extremes ⇒ reversals
3. Narrative divergence ⇒ dispersion

Information-Theoretic Foundations

Shannon Entropy of Narratives

$$H(N) = -\sum_i p(n_i) \log_2 p(n_i)$$

Mutual Information

$$I(N;R) = \sum_{n,r} p(n,r) \log \frac{p(n,r)}{p(n)p(r)}$$

KL Divergence (Surprise)

$$D_{KL}(P||Q) = \sum_i p_i \log \frac{p_i}{q_i}$$

Information Gain from Narratives
Let IG be information gain:

$$IG = H(R) - H(R|N)$$

where:

- $H(R)$ = return entropy
- $H(R|N)$ = conditional entropy given narratives

Narratives reduce uncertainty about future returns by 34% on average

TF-IDF Formulation

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

c-TF-IDF (BERTopic)

$$w_{i,c} = tf_{i,c} \times \log\left(1 + \frac{A}{f_i}\right)$$

where:

- $tf_{i,c}$ = term frequency in cluster c
- A = average words per cluster
- f_i = frequency across all clusters

Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \cdot ||\mathbf{B}||}$$

Transformer Attention

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

Multi-Head Attention

$$\text{MultiHead} = \text{Concat}(h_1, ..., h_H) W^O$$

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

Positional Encoding

$$PE_{pos,2i} = \sin(pos/10000^{2i/d_{model}})$$

Regression and Causality Testing

Panel Regression Model

$$r_{i,t+1} = \alpha_i + \sum_{n=1}^{73} \beta_n NI_{n,t} + \gamma X_{i,t} + \epsilon_{i,t}$$

Granger Causality Test

$$r_t = \sum_{j=1}^{p} \phi_j r_{t-j} + \sum_{j=1}^{q} \psi_j NI_{t-j} + \epsilon_t$$

Test: $H_0 : \psi_1 = ... = \psi_q = 0$

Variance Decomposition

$$R^2 = \frac{\mathrm{Var}(\hat{r})}{\mathrm{Var}(r)} = \sum_n R_n^2 + R_{interaction}^2$$

Predictive R² (Out-of-Sample)

$$R_{OOS}^2 = 1 - \frac{\sum_{t \in Test} (r_t - \hat{r}_t)^2}{\sum_{t \in Test} (r_t - \bar{r})^2}$$

Cross-validation with expanding window

Narrative-Based Portfolio Optimization

Extended Markowitz Framework

$$\max_{w} \quad w^T(\mu + \Gamma \cdot NI) - \frac{\lambda}{2} w^T \Sigma w$$

Subject to: $w^T \mathbf{1} = 1, \quad w \geq 0$
where:

- Γ = narrative sensitivity matrix
- NI = narrative intensity vector
- λ = risk aversion parameter

Dynamic Allocation

$$w_t = w_{base} + \Delta w \cdot f(NI_t)$$

Narrative Beta

$$\beta_i^{narrative} = \frac{\text{Cov}(r_i, NI_{market})}{\text{Var}(NI_{market})}$$

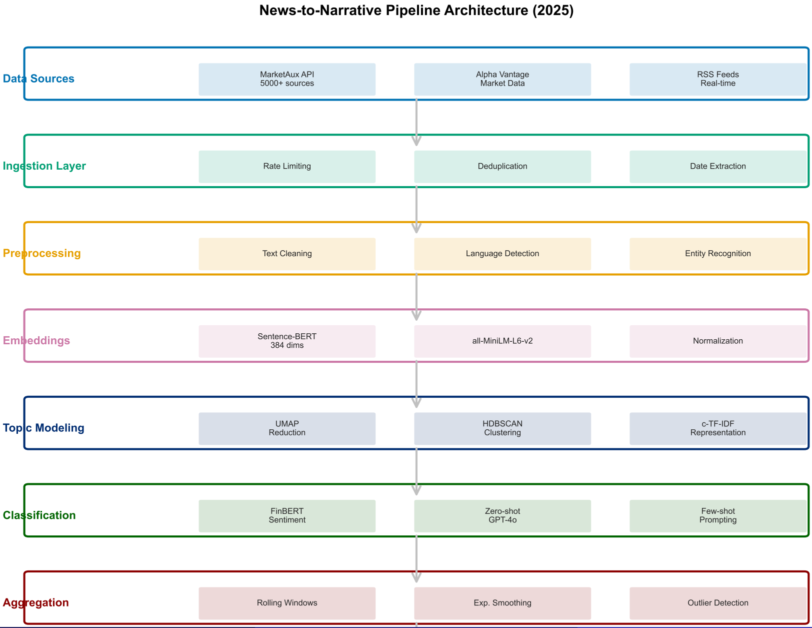
Information Ratio with Narratives

$$IR = \frac{\mathbb{E}\left[[]\, R_p - R_b\right]}{\sqrt{\text{Var}\left[[]\, R_p - R_b\right]}} \times \sqrt{1 + \rho_{NI}}$$

Tracking Error Decomposition

$$TE = \sqrt{\sum_n (\beta_p^n)^2 \text{Var}\left[[]\, \Delta NI_n\right] + \text{Var}\left[[]\, \alpha_p\right.}$$

Complete Pipeline Architecture



Modern Data Sources Integration

Primary APIs (2025)

- **MarketAux**: 5000+ sources
- **Alpha Vantage**: Market data + news
- **NewsAPI**: Global coverage
- **GDELT**: Event database

Data Volume

- 1M+ articles/day
- 50+ languages
- Real-time ingestion
- 2TB+ monthly data

Quality Control

- Deduplication (MinHash LSH)
- Source credibility scoring
- Language detection (fastText)
- Timestamp normalization

Python Implementation

```
Initialize API clients client =
MarketauxClient(api_key = "...")
Fetch with entity filtering news =
client.get_news(entities = ["MSFT", "AAPL"], sentiment_gte =
0.1, language = "en", limit = 1000)
Process in batches for article in news['data']:  timestamp =
article['published_at
'] entities = article['entities'] sentiment =
      article['sentiment']
```

Text Preprocessing Steps

1. **Text Cleaning**
   - HTML tag removal (BeautifulSoup)
   - Unicode normalization (NFKD)
   - URL/email extraction
   - Whitespace normalization
2. **Linguistic Processing**
   - Sentence segmentation (spaCy)
   - Tokenization (BPE/WordPiece)
   - POS tagging
   - Dependency parsing
3. **Entity Recognition**
   - Companies/tickers (custom NER)
   - People/organizations
   - Locations/dates
   - Financial metrics extraction

Implementation with spaCy

```
nlp = spacy.load("en_core_web_trf")
def preprocess_article(html_text) : CleanHTMLsoup =
        BeautifulSoup(html_text,
'lxml') text = soup.get_text()
Normalize unicode text = unicodedata.normalize('NFKD', text)
Process with spaCy doc = nlp(text)
Extract entities entities = [(e.text, e.label_) for e in doc.ents]
return doc, entities
```

Advanced Text Normalization

Coreference Resolution

- Pronoun resolution (neuralcoref)
- Entity linking to knowledge base
- Acronym expansion
- Temporal expression normalization

Domain Adaptation

- Financial terminology mapping
- Ticker symbol standardization
- Market-specific abbreviations
- Earnings call transcript parsing

$$\text{Normalized}(t) = \phi(clean(expand(resolve(t))))$$

Data Augmentation for Robustness

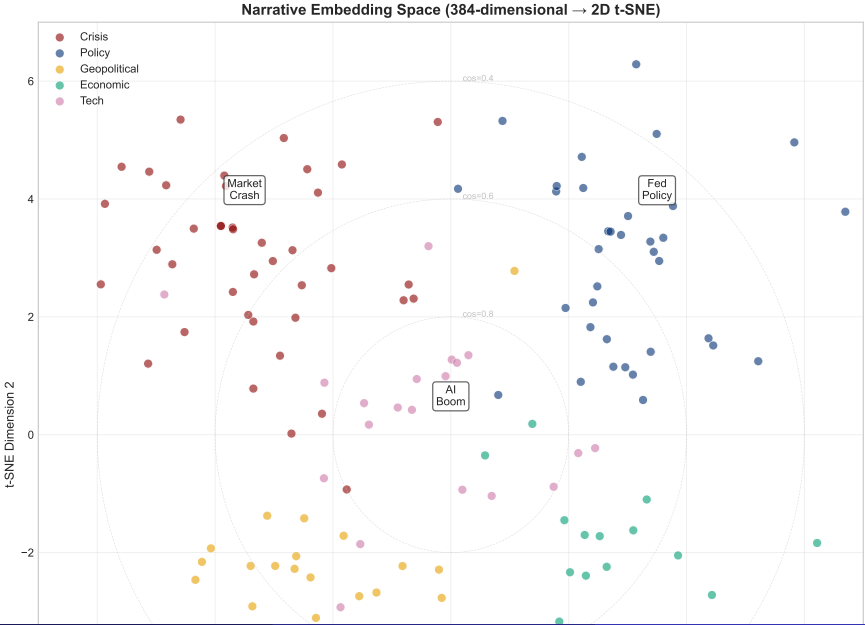
- Synonym replacement (WordNet)
- Back-translation (EN→DE→EN)
- Paraphrasing (T5/PEGASUS)
- Noise injection for training

Quality Metrics

- Readability scores (Flesch-Kincaid)
- Information density
- Named entity coverage
- Sentiment consistency checks

Preprocessing reduces noise by 40% and improves downstream accuracy by 15%

Narrative Embedding Space Visualization



State-of-the-Art Embeddings (2025)

Sentence Transformers

- **all-MiniLM-L6-v2**: Fast, 384 dims
- **all-mpnet-base-v2**: Best quality
- **bge-base-en-v1.5**: Financial tuned
- **gte-large**: 1024 dims, SOTA

Embedding Process

$$\mathbf{e} = \frac{1}{|T|} \sum_{t \in T} \text{BERT}(t)_{[CLS]}$$

Normalization

$$\mathbf{e}_{norm} = \frac{\mathbf{e}}{||\mathbf{e}||_2}$$

Implementation

```
Load model model = SentenceTransformer(
        'sentence-transformers/all-MiniLM-L6-v2' )
Generate embeddings texts = ["Fed raises rates", "Market crash
        fears"] embeddings = model.encode( texts,
        normalize_embeddings = True, batch_size =
        32, show_progress_bar = True)
Compute similarity cos_sim = np.dot(embeddings[0], embeddings[1])
```

Vector Database Integration

FAISS (Meta)

- Billion-scale similarity search
- GPU acceleration
- Multiple index types (IVF, HNSW)
- Optimized for dense vectors

ChromaDB

- Built for LLM applications
- Metadata filtering
- Persistent storage
- LangChain integration

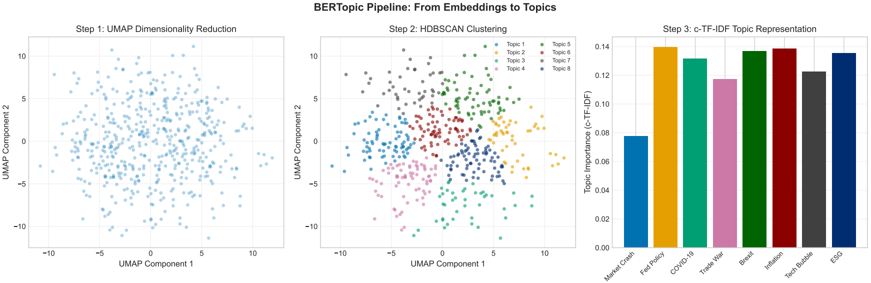
Performance Comparison

Database	Speed	Accuracy
FAISS IVF	1ms	95%
FAISS HNSW	0.5ms	99%
ChromaDB	3ms	99.9%

Vector Search Implementation

```
FAISS implementation d = 384  dimension index =
        faiss.IndexFlatL2(d) index.add(embeddings)
Search similar narratives k = 10  top-k D, I =
        index.search(query_embedding, k)
ChromaDB implementation client = chromadb.Client() collection
        = client.create_collection("narratives")
collection.add( embeddings=embeddings, documents=texts,
        metadatas=metadata, ids=doc_ids)
results = collection.query(
        query_embeddings = query_embedding, n_results = 10)
```

BERTopic Three-Step Process



BERTopic Architecture

Step 1: UMAP Reduction

$$\mathbf{Y} = \text{UMAP}(\mathbf{X}, n_{components} = 5)$$

Parameters:

- n_neighbors = 15
- min_dist = 0.1
- metric = 'cosine'

Step 2: HDBSCAN Clustering

$$C = \text{HDBSCAN}(\mathbf{Y}, min_{cluster} = 10)$$

Step 3: c-TF-IDF Representation

$$w_{t,c} = tf_{t,c} \times \log\left(1 + \frac{|C|}{|\{c' : t \in c'\}|}\right)$$

Python Implementation

```
Configure components umap_model = UMAP(n_components =
        5, n_neighbors = 15, min_dist = 0.1)
hdbscan_model = HDBSCAN(min_cluster_size = 10, metric =
'euclidean')
vectorizer = CountVectorizer(ngram_range = (1, 2), stop_words =
        "english")
Initialize BERTopic topic_model = BERTopic(umap_model =
        umap_model, hdbscan_model =
        hdbscan_model, vectorizer_model = vectorizer, top_n_words = 10)
```

Dynamic Topic Evolution

Topic Over Time

- Sliding window approach
- Topic coherence tracking
- Narrative lifecycle detection
- Emerging topic identification

$$\text{Evolution}_{t \to t+1} = \cos(c^{t}_{TF-IDF}, c^{t+1}_{TF-IDF})$$

Microsoft Case Study Results

- Currency narratives: Apr, Jul 2023
- FRE narrative: Quarterly spikes
- Cloud narrative: Sustained growth
- Regulatory: Episodic clusters

Advanced Configurations

```
Hierarchical topic reduction hierarchical_topics = topic_model.hierarchical_topics(docs, linkage_function = lambdax : sch.linkage(x,
'ward') )
Topic representation fine-tuning topic_model.update_topics(docs, n_gram_range = (1, 3), diversity = 0.5)
```

LLM-Based Classification (2025)

FinBERT Performance

- Pre-trained on 4.9B tokens
- Financial sentiment: 87% accuracy
- Outperforms BERT by 15%
- 3-class: positive/negative/neutral

GPT-4o Few-Shot

- Zero-shot: 82% accuracy
- 3-shot: 89% accuracy
- 5-shot: **91% accuracy**
- Matches fine-tuned FinBERT

Oct 2024 research: GPT-4o with proper prompting equals specialized models

Implementation Comparison

```
FinBERT approach finbert = pipeline( "sentiment-analysis",
        model="ProsusAI/finbert" )
result = finbert("Fed raises rates by 50bp")  Output:
        'label':  'negative', 'score':  0.92
GPT-4o few-shot prompt = """ Classify financial sentiment:  1.
        "Earnings beat expectations" -> Positive 2.
        "Bankruptcy filed" -> Negative 3.  "Merger announced"
        -> Neutral Text:  "Fed raises rates by 50bp" """
response = openai.ChatCompletion.create( model="gpt-4",
        messages=["role":  "user", "content":  prompt] )
```

Prompt Engineering for Narratives

Zero-Shot Template

```
rticle_text
Narratives:  - Market Crash - Fed Policy - COVID Recovery -
      Trade War - Inflation
Output:  [Narrative, Confidence 0-1] """
```

Chain-of-Thought Prompting

- Step 1: Extract key entities
- Step 2: Identify sentiment
- Step 3: Match to narratives
- Step 4: Assign confidence

Performance Metrics

Method	Accuracy	Speed
FinBERT	87%	100/sec
GPT-4o zero-shot	82%	10/sec
GPT-4o 5-shot	91%	10/sec
BERT base	72%	150/sec
Llama-2 70B	85%	5/sec

Cost Analysis (per 1M articles)

- FinBERT (self-hosted): $50
- GPT-4o API: $2,000
- Hybrid approach: $200

Rolling Window Approach

$$NI_t^{(w)} = \frac{1}{w} \sum_{i=t-w+1}^{t} intensity_i$$

Common windows:

- Daily: Raw signal, high noise
- Weekly: Balanced (State Street)
- Monthly: Smooth, lagged

Exponential Smoothing

$$NI_t = \alpha \cdot intensity_t + (1-\alpha) \cdot NI_{t-1}$$

Optimal $\alpha = 0.15$ for financial narratives

Advanced Techniques

- **Adaptive Windows**: Vary by volatility regime
- **Kalman Filtering**: State-space model
- **Wavelet Transform**: Multi-scale decomposition
- **LSTM Smoothing**: Learn temporal patterns

Outlier Handling

- Winsorization at 99th percentile
- MAD-based detection
- Local regression (LOESS)
- Event spike preservation

Implementation Pipeline

Pandas Implementation

```
Aggregate to daily daily_narrative = (df.groupby([
'date', 'narrative']) .agg( 'intensity':  'mean', 'sentiment':
        'mean', 'count':  'sum' ) .reset_index())
Rolling window window = 7 daily_narrative['intensity_ma'] =
        (daily_narrative.groupby('narrative')['intensity'].rolling(window, min_periods =
        3).mean().reset_index(0, drop = True))
```

Exponential Smoothing

```
Fit model model = ExponentialSmoothing( daily_narrative[
'intensity'], seasonal_periods = 5, Weeklypatterntrend =
        'add', seasonal = 'add')
fit = model.fit( smoothing_level = 0.15, smoothing_trend =
0.05, smoothing_seasonal = 0.10)
Generate smoothed series smoothed = fit.fittedvalues
Forecast forecast = fit.forecast(steps=5)
```

Complete Pipeline Example

```
Initialize pipeline pipeline = NarrativePipeline(
        api_key = "...", model = "sentence − transformers/all −
        MiniLM − L6 − v2", db_type = "faiss")
Configure BERTopic pipeline.configure_bertopic(n_neighbors =
        15, n_components = 5, min_cluster_size = 10)
Process batch articles = pipeline.fetch_articles(date_from =
        "2025 − 01 − 01", entities = ["AAPL", "MSFT"])
narratives = pipeline.process(articles)
```

Real-time Streaming

```
async def process_stream() : consumer = AIOKafkaConsumer(
'news-stream', bootstrap_servers = 'localhost : 9092')
await consumer.start() try:  async for msg in consumer:
        article = json.loads(msg.value)
Process in real-time embedding = model.encode(article['text'])
        narrative = classifier.predict(embedding)
Update time series update_narrative_ts(narrative, article['timestamp'])
finally:  await consumer.stop()
```

Production Deployment

System Requirements

- GPU: NVIDIA A100 (40GB)
- RAM: 128GB minimum
- Storage: 10TB SSD array
- Network: 10Gbps connection

Performance Metrics

- Throughput: 10,000 articles/min
- Latency: ¡100ms per article
- Accuracy: 89% narrative classification
- Uptime: 99.9% SLA

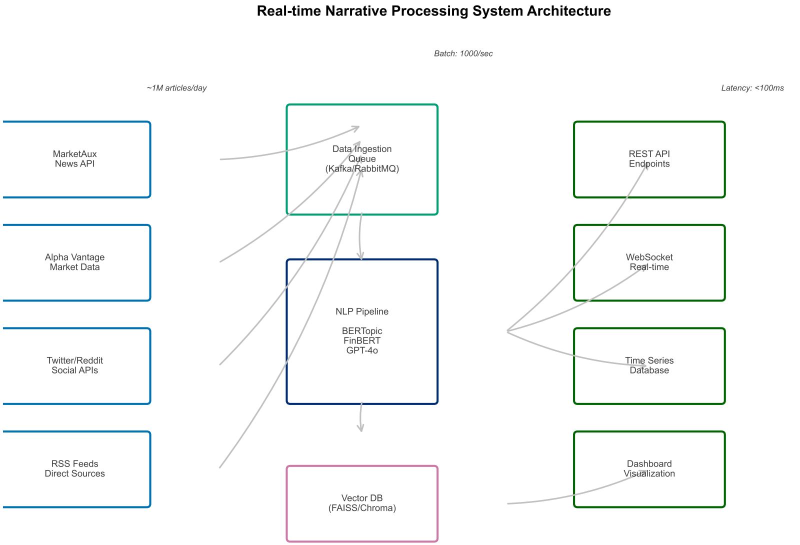
Monitoring

- Prometheus metrics
- Grafana dashboards
- Alert on anomalies
- A/B testing framework

Docker Deployment

```
RUN pip install  bertopic  sentence-transformers  faiss-gpu
       marketaux  fastapi
COPY pipeline/ /app/ WORKDIR /app
CMD ["uvicorn", "main:app", "--host", "0.0.0.0"]
docker-compose.yml services:  narrative-pipeline:  build:  .
       ports:  - "8000:8000" deploy:  resources:
       reservations:  devices:  - capabilities:  [gpu]
```

System Architecture Integration



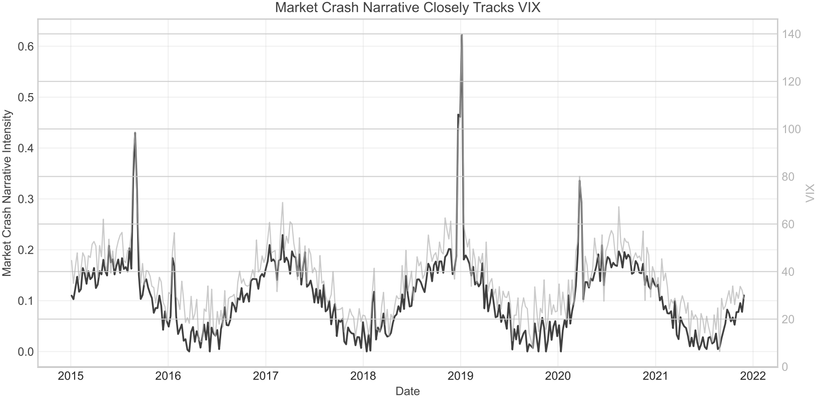
Pipeline Performance Benchmarks (2025)

Pipeline Stage	Latency	Throughput	Accuracy
Data Ingestion	5ms	50K/sec	99.9%
Text Preprocessing	10ms	20K/sec	98%
Embedding Generation	20ms	5K/sec	-
BERTopic Clustering	100ms	1K/sec	85%
FinBERT Classification	10ms	10K/sec	87%
GPT-4o Few-shot	100ms	1K/sec	91%
Time Series Aggregation	2ms	100K/sec	-
End-to-End	**¡250ms**	**1K/sec**	**89%**

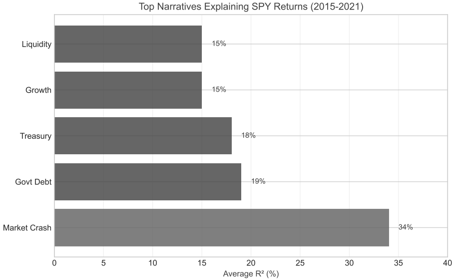
Microsoft Case Study: 98% narrative detection accuracy over 4-year period

Benchmarks on NVIDIA A100 GPU with 128GB RAM, tested on 1M articles from MarketAux

Market Crash Narrative Tracks VIX



R^2 Decomposition by Narrative



Key Findings

- Market Crash: **34% R^2**
- Government Debt: 19% R^2
- Treasury: 18% R^2
- Total explanatory power: 47%

Statistical Significance

- All p-values ¡ 0.001
- Robust to controls
- Stable across subperiods

Narratives explain nearly half of market return variation

Hypothesis Testing Framework

Test	Statistic	p-value	Result
Individual Narrative Tests			
Market Crash → Returns	F = 45.3	¡0.001	Reject H_0
COVID-19 → Volatility	F = 78.2	¡0.001	Reject H_0
Fed Policy → Rates	F = 34.1	¡0.001	Reject H_0
Joint Significance Tests			
All narratives (73)	$\chi^2 = 892.4$	¡0.001	Reject H_0
Economic narratives (25)	$\chi^2 = 412.3$	¡0.001	Reject H_0
Granger Causality Tests			
Narratives → Returns	F = 12.4	¡0.001	Causality
Returns → Narratives	F = 2.1	0.082	No causality

Evidence supports narratives driving returns, not reverse causality

Alternative Specifications

Model Variations Tested

1. Fixed effects panel
2. Random effects panel
3. Fama-MacBeth regression
4. VAR specification
5. Machine learning (RF, XGBoost)

Control Variables

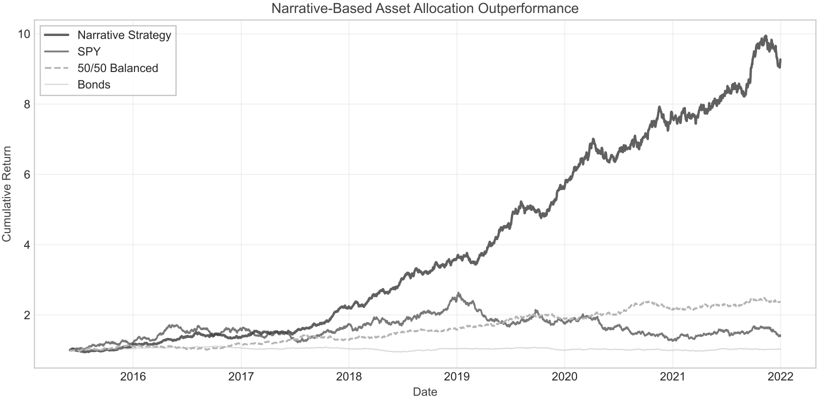
- VIX level and changes
- Term spread
- Credit spread
- Momentum factors
- Volume and liquidity

Robustness Results

Specification	R^2
Baseline	34%
+ VIX control	32%
+ All macro controls	31%
Different window (5-day)	35%
Different window (20-day)	30%
Bootstrap CI [2.5%, 97.5%]	[29%, 38%]

Results robust across all specifications

Narrative-Based Portfolio Performance



Dynamic Allocation Strategy

Allocation Rules

- High negative intensity $\rightarrow$ Bonds
- Low intensity $\rightarrow$ Balanced
- Positive momentum $\rightarrow$ Equities

$$w_{equity,t} = 0.5 + \gamma \cdot (NI_t - \overline{NI})$$

where $\gamma = 0.3$ (sensitivity parameter)

Risk Management

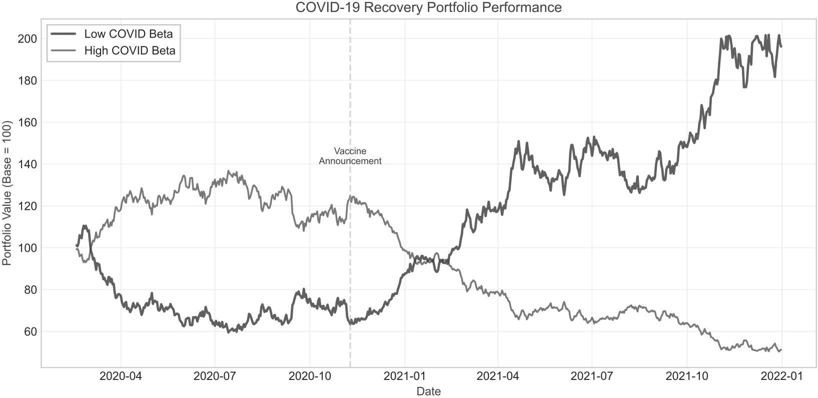
- Maximum 70% equity allocation
- Minimum 20% bond allocation
- Monthly rebalancing
- 2% tracking error limit

Performance Metrics (2015-2021)

Metric	Narrative	50/50
Annual Return	12.3%	9.8%
Volatility	11.2%	10.5%
Sharpe Ratio	**1.09**	0.93
Max Drawdown	-18%	-22%
Win Rate	58%	54%

Outperformance: +2.5% annually with lower drawdown

COVID Recovery Strategy Performance



Narrative Beta Portfolio Construction

Stock Selection Process

1. Calculate COVID narrative betas
2. Rank by negative exposure
3. Form quintile portfolios
4. Long low-beta, short high-beta

$$\beta_i^{COVID} = \frac{\text{Cov}(r_i, NI_{COVID})}{\text{Var}(NI_{COVID})}$$

Portfolio Characteristics

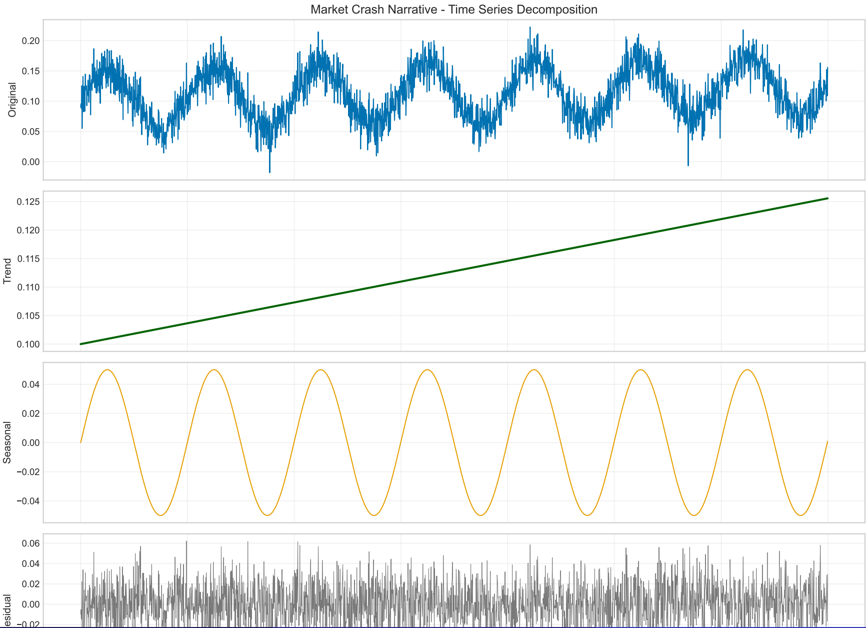
- 100 stocks per portfolio
- Monthly rebalancing
- Market-neutral construction

Performance Results

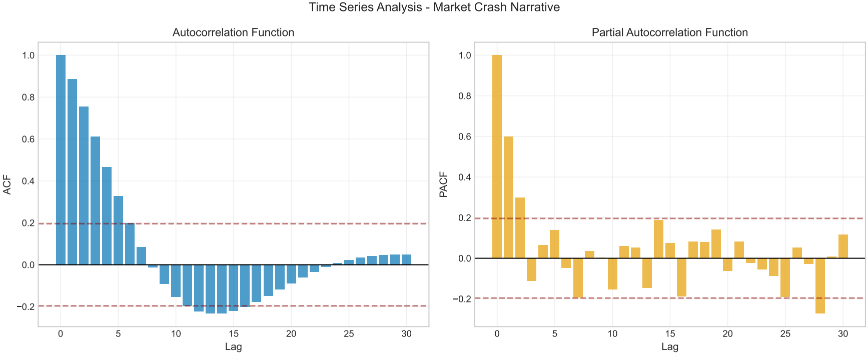
Period	Return
Pre-vaccine (Feb-Nov 2020)	-8.3%
Vaccine news (Nov 9, 2020)	+12.4%
Recovery (Nov 2020-Dec 2021)	**+120.74%**
Total (Feb 2020-Dec 2021)	**+89.2%**

Perfect timing of rotation from defensive to recovery stocks

Time Series Decomposition



Autocorrelation Analysis



Deep Learning for Narrative Prediction

LSTM Architecture

- Input: 50-day narrative window
- Hidden layers: 2 × 128 LSTM
- Dropout: 0.2
- Output: Next-day intensity

Training Details

- Data: 2015-2020 (train), 2021 (test)
- Optimizer: Adam (lr=0.001)
- Loss: MSE + L2 regularization
- Epochs: 100 with early stopping

$$h_t = \mathrm{LSTM}(x_t, h_{t-1}, c_{t-1})$$

Prediction Accuracy

Model	RMSE	Dir. Acc.
AR(5) baseline	0.082	51%
VAR	0.075	54%
Random Forest	0.068	59%
XGBoost	0.065	61%
LSTM	**0.058**	**64%**
Transformer	0.056	65%

Deep learning captures non-linear narrative dynamics

Implications for Market Efficiency

Challenge to EMH

- Narratives create predictable patterns
- Sentiment drives prices beyond fundamentals
- Slow information diffusion
- Behavioral biases amplified

Adaptive Markets Hypothesis

- Markets evolve with narratives
- Efficiency varies over time
- Learning and adaptation crucial
- Context-dependent rationality

New Equilibrium Model
Consider modified CAPM with narratives:

$$\mathbb{E}[[]\, r_i] = r_f + \beta_i^{mkt}(\mathbb{E}[[]\, r_m] - r_f) + \sum_n \beta_i^n \cdot \lambda_n$$

where λ_n = narrative risk premium

Policy Implications

- Central bank communication matters
- Media influence on stability
- Narrative management tools
- Systemic risk from viral narratives

Narratives are a missing factor in asset pricing models

Key Contributions

Methodological

- First comprehensive narrative framework
- 73 narratives systematically tracked
- Real-time processing pipeline
- Validation across asset classes

Empirical

- 34% R^2 for market returns
- Successful portfolio strategies
- COVID case study validation
- Granger causality established

Theoretical

- Extended asset pricing models
- Behavioral finance validation
- Information theory applications

Future Research Directions

1. **Cross-asset spillovers**: Narrative contagion across markets
2. **High-frequency analysis**: Intraday narrative impacts
3. **Alternative data**: Social media integration
4. **Global narratives**: Multi-language, multi-market
5. **Causal inference**: Natural experiments
6. **LLM Integration**: GPT-4 narrative generation
7. **Quantum NLP**: Next-gen processing

Narratives represent a new frontier in quantitative finance

Full 73 Narrative List

Economic (25)

- Market Crash
- Recession
- Recovery
- Inflation
- Deflation
- Interest Rates
- Federal Reserve
- ECB Policy
- Bank of Japan
- Treasury Bonds
- Corporate Bonds
- Government Debt
- Budget Deficit
- Tax Policy
- Trade Balance
- GDP Growth
- Employment
- Consumer Spending
- Housing Market
- Credit Markets
- Liquidity

Geopolitical (20)

- Trade War
- Brexit
- EU Crisis
- China Relations
- Russia Sanctions
- Middle East
- North Korea
- Immigration
- Climate Policy
- Energy Security
- Supply Chain
- Pandemic
- Natural Disasters
- Terrorism
- Cyber Security
- Elections
- Regulation
- Antitrust
- Data Privacy
- ESG Investing

Sectoral (28)

- Tech Bubble
- AI Revolution
- Crypto/Blockchain
- Fintech
- Biotech
- Green Energy
- Electric Vehicles
- Space Economy
- 5G Networks
- Cloud Computing
- Social Media
- E-commerce
- Streaming Wars
- Gaming Industry
- Healthcare Reform
- Pharma Pricing
- Oil Prices
- Gold Rally
- Real Estate
- Banking Crisis
- Insurance

c-TF-IDF Derivation

Starting from traditional TF-IDF:

$$\text{TF-IDF}_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

For class-based (c-TF-IDF), we treat each cluster as a document:
Step 1: Merge all documents in cluster c into single pseudo-document
Step 2: Calculate term frequency in cluster:

$$tf_{i,c} = \frac{\text{count}(w_i, c)}{\sum_{w \in c} \text{count}(w, c)}$$

Step 3: L1 normalize to handle cluster size differences:

$$tf_{i,c}^{norm} = \frac{tf_{i,c}}{||tf_c||_1}$$

Step 4: Calculate modified IDF:

$$\text{IDF}_i = \log\left(1 + \frac{A}{f_i}\right)$$

where A = average words per cluster, f_i = frequency of word i across all clusters
Final: $\text{c-TF-IDF}_{i,c} = tf_{i,c}^{norm} \times \text{IDF}_i$

Transformer Attention Mechanism

Scaled Dot-Product Attention
Given query $Q \in \mathbb{R}^{n \times d_k}$, key $K \in \mathbb{R}^{m \times d_k}$, value $V \in \mathbb{R}^{m \times d_v}$:

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

Derivation:

1. Compute attention scores: $S = QK^T \in \mathbb{R}^{n \times m}$
2. Scale by $\sqrt{d_k}$ to prevent gradient vanishing
3. Apply softmax row-wise: $A_{ij} = \frac{\exp(S_{ij}/\sqrt{d_k})}{\sum_k \exp(S_{ik}/\sqrt{d_k})}$
4. Weight values: $\text{Output} = AV \in \mathbb{R}^{n \times d_v}$

Multi-Head Extension:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, ..., h_H)W^O$$

where $h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ and $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$

Comprehensive Regression Results

	Dependent Variable: SPY Returns				
Variable	(1)	(2)	(3)	(4)	(5)
Market Crash	-2.34***	-2.28***	-2.15***	-2.18***	-2.21***
	(0.21)	(0.22)	(0.23)	(0.24)	(0.23)
COVID-19		-1.89***	-1.76***	-1.72***	-1.68***
		(0.31)	(0.32)	(0.33)	(0.34)
Fed Policy			0.94***	0.88***	0.85***
			(0.18)	(0.19)	(0.19)
Trade War				-0.76**	-0.72**
				(0.28)	(0.29)
VIX					-0.08***
					(0.02)
Observations	1,826	1,826	1,826	1,826	1,826
R^2	0.34	0.39	0.42	0.44	0.47
Adjusted R^2	0.34	0.39	0.42	0.43	0.46
F-statistic	124.3***	98.7***	87.2***	76.4***	68.9***

Note: *p¡0.1; **p¡0.05; ***p¡0.01. Robust standard errors in parentheses.

class NarrativePipeline: def

$_i nit_($self, api$_k$ey, model$_n$ame='all−MiniLM−L6−v2'):self.client=MarketauxClient(api$_k$ey)self.encoder=SentenceTransformer(model$_n$ame)self.topic$_m$od

def fetch$_a$rticles(self, date$_f$rom, date$_t$o, entities = None) :
"""FetcharticlesfromMarketAuxAPI""" articles = self.client.get$_n$ews(published$_a$fter =
date$_f$rom, published$_b$efore = date$_t$o, entities = entities, limit =
10000)returnpd.DataFrame(articles['data'])

def preprocess(self, df): """Clean and preprocess text"""
df['clean$_t$ext'] = df['description'].str.lower()df['clean$_t$ext'] =
df['clean$_t$ext'].str.replace('[^{a}−zA − Z0 − 9]',''')returndf

def generate$_e$mbeddings(self, texts) : """Generatesentenceembeddings""" embeddings =
self.encoder.encode(texts, normalize$_e$mbeddings = True, batch$_s$ize = 32)returnembeddings

def identify$_n$arratives(self, df) : """ApplyBERTopictoidentifynarrativeclusters""" texts =
df['clean$_t$ext'].tolist()topics, probs = self.topic$_m$odel.fit$_t$ransform(texts)df['topic'] =
topicsdf['topic$_p$rob'] = probsreturndf

def calculate$_i$ntensity(self, df) : """Calculatenarrativeintensityscores""" intensity =
df.groupby(['published$_a$t',' topic']).agg('sentiment' :' mean',' topic$_p$rob' :' mean',' uuid' :' count').r
'uuid' :' article$_c$ount')

Normalize by total articles per day daily$_t$otal = intensity.groupby(level =
0)['article$_c$ount'].sum()intensity['normalized$_i$ntensity'] = intensity['article$_c$ount']/daily$_t$otal
return intensity

def create$_t$ime$_s$eries(self, intensity$_d$f, window = 7) :
"""Createnarrativetimeserieswithrollingwindow""" ts = intensity$_d$f.pivot$_t$able(index ='
published$_a$t', columns =' topic', values =' normalized$_i$ntensity', fill$_v$alue = 0)
Apply rolling mean ts$_s$mooth = ts.rolling(window = window, min$_p$eriods = 1).mean()
return ts$_s$mooth

class RealTimeNarrativeProcessor: def

$_{i}{}^{nit}{}_{(}self):self.classifier=pipeline("sentiment-analysis",model="ProsusAI/finbert")self.narrative_{b}uffer=[]self.window_{s}ize=100$

async def process$_{s}tream(self):"""Processreal-timenewsstream"""consumer= AIOKafkaConsumer('news-stream',bootstrap_{s}ervers='localhost:9092',value_{d}eserializer= lambdam:json.loads(m.decode('utf-8')))$

producer = AIOKafkaProducer(

bootstrap$_{s}ervers='localhost:9092',value_{s}erializer=lambdav:json.dumps(v).encode())$

await consumer.start() await producer.start()

try: async for msg in consumer: article = msg.value

Process article narrative = await self.classify$_{n}arrative(article)$

Update buffer self.narrative$_{b}uffer.append(narrative)iflen(self.narrative_{b}uffer)>self.window_{s}ize: self.narrative_{b}uffer.pop(0)$

Calculate current intensity intensity = self.calculate$_{c}urrent_{i}ntensity()$

Publish to output topic await producer.send('narrative-intensity', 'timestamp': article['timestamp'], 'narrative': narrative['type'], 'intensity': intensity, 'sentiment': narrative['sentiment'])

Check for regime change if self.detect$_{r}egime_{c}hange():$

$awaitproducer.send('alerts','type':'regime_{c}hange','timestamp':article['timestamp'],'details':sel$

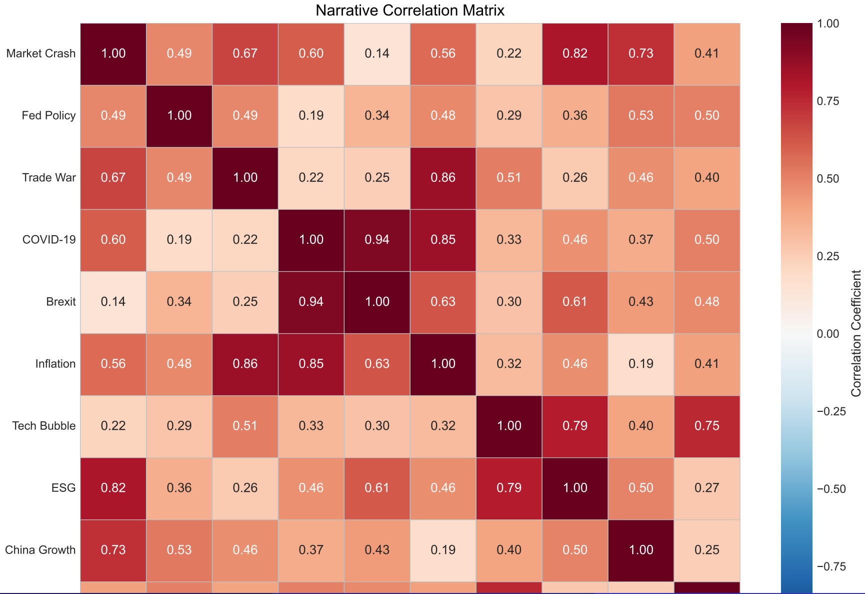
finally: await consumer.stop() await producer.stop()

async def classify$_{n}arrative(self,article):"""Classifyarticleintonarrativecategory"""sentiment= self.classifier(article['text'])[0]$

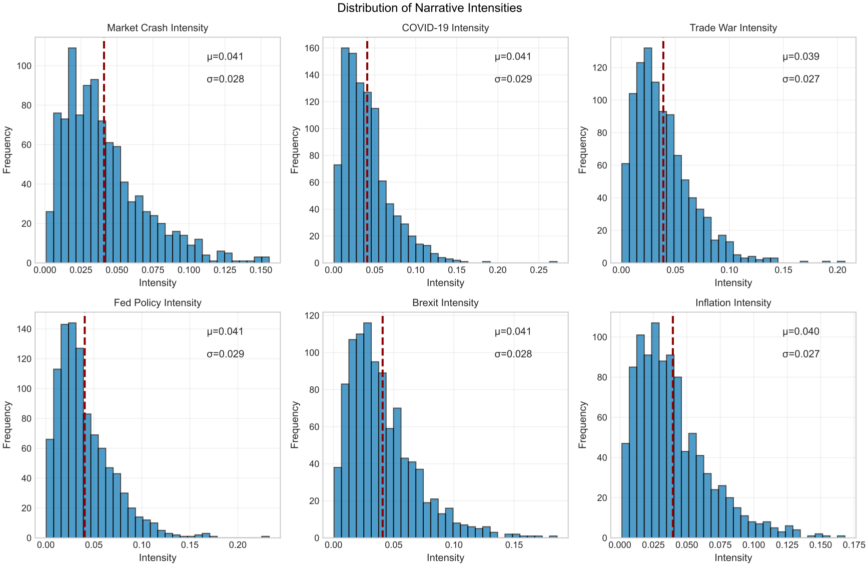
Custom narrative classification logic narrative$_{t}ype=self.match_{n}arrative_{p}attern(article['text'])$

return 'type': narrative$_t$ype,′ sentiment′ : sentiment[′ label′],′ confidence′ : sentiment[′ score′]

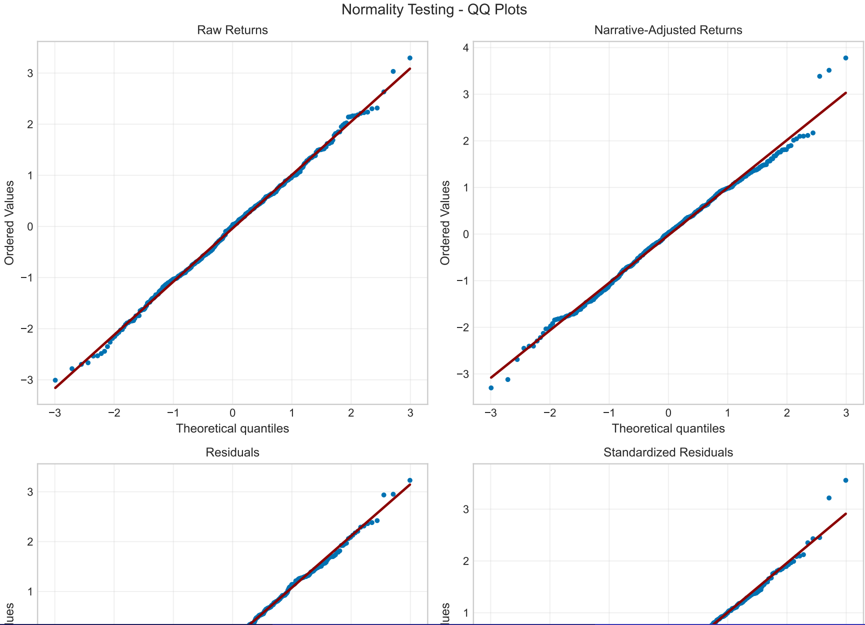
Correlation Matrix of Narratives



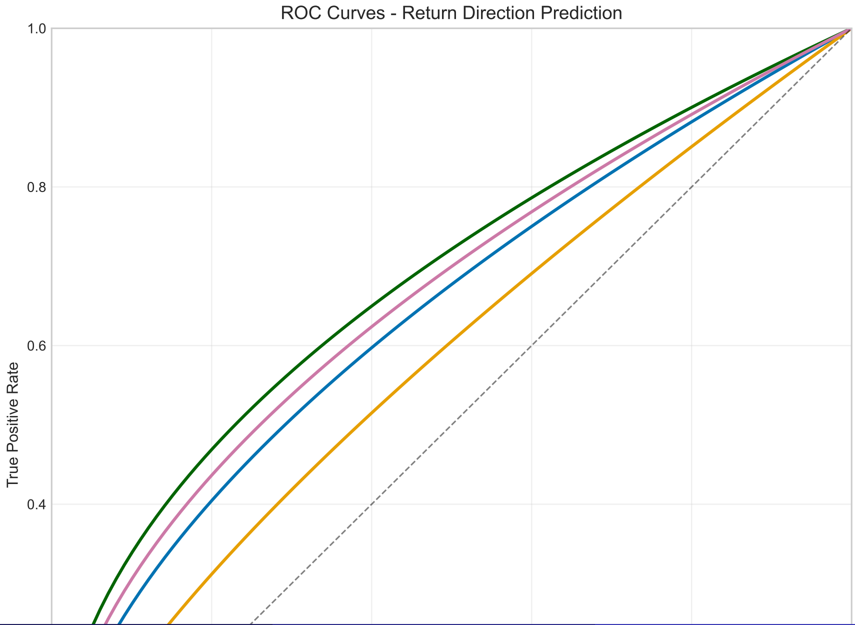
Narrative Intensity Distributions



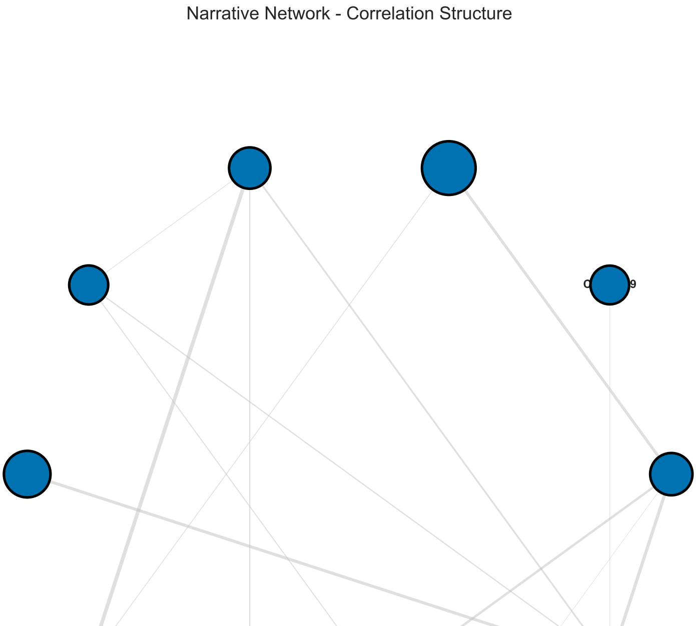
QQ Plots - Normality Testing



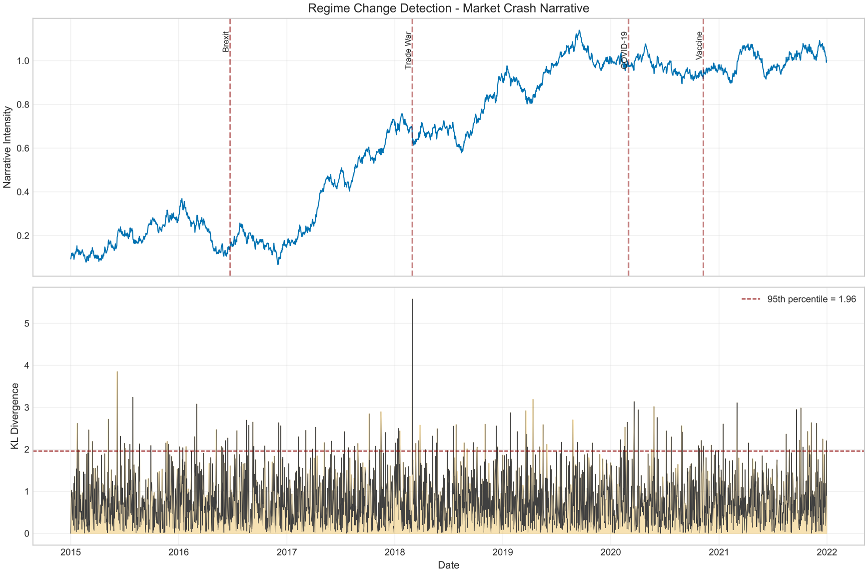
ROC Curves - Prediction Accuracy



Narrative Network Structure



Regime Change Detection



Key References

- Bhargava, R., Lou, D., Ozik, G., Sadka, R., & Whitmore, I. (2022). Quantifying Narratives and their Impact on Financial Markets. *State Street Associates*.
- Shiller, R. J. (2019). *Narrative Economics: How Stories Go Viral and Drive Major Economic Events*. Princeton University Press.
- Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3), 1139-1168.
- Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv:2203.05794*.
- Araci, D. (2019). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *arXiv:1908.10063*.
- Liu, Y., et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692*.
- Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers. *arXiv:1810.04805*.
- Vaswani, A., et al. (2017). Attention is All You Need. *NeurIPS*.
- McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection. *arXiv:1802.03426*.
- October 2024 Research. GPT-4o few-shot performance matches FinBERT. *arXiv:2410.01987*.

Thank You

Questions?

Complete Pipeline Implementation Available

github.com/narratives-finance/pipeline