Market Mood & Moves
Sentiment-Driven Stock Prediction

Week

Mentors: Meet & Sarthak

December 2025

Welcome to Week 2

Last week, we treated FinBERT as a "black box"—a magic tool that took text and gave us sentiment.
This week, we open the box. To build a truly robust trading system, you must understand the architecture that powers your signals. We will move from the limitations of static embeddings (Word2Vec) to the dynamic attention mechanisms of BERT, and finally, into the specific domain adaptation techniques that make **FinBERT** superior for financial markets.
Key Themes:

- **Contextual vs. Static:** Why "Bank" isn't just a building.
- **The Transformer:** Self-Attention mathematics.
- **Domain Adaptation:** How to teach a generic model to speak "Finance".

Contents

1 The Evolution of Embeddings

The history of Natural Language Processing is the history of trying to represent words as numbers while preserving their meaning.

1.1 The Limitation of Static Embeddings

Models like Word2Vec and GloVe map each word in the vocabulary to a single, static vector $\mathbf{v}_w \in \mathbb{R}^d$. This mapping is constant regardless of context.

This approach suffers fundamentally from the problem of **Polysemy**—the capacity for a word to have multiple related meanings.

Real-Life Example: The "Bank" Problem

Consider the word "Bank":

1. "I deposited money at the **bank**." (Financial Institution)
2. "They sat on the river **bank**." (Geological Feature)

In a static model (Word2Vec), the vector $\mathbf{v}_{\text{bank}}$ is a weighted average of these two disparate meanings. It is a "smeared" representation that is neither fully financial nor fully geological.

1.2 Contextual Representations

BERT (Bidirectional Encoder Representations from Transformers) introduced the concept of dynamic embeddings.

Insight: Embeddings as Functions

In static models, an embedding is a **Lookup Table**. In BERT, an embedding is a **Function**.

$$\mathbf{v}_{\text{BERT}}(w_i) = f(w_i \mid w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n; \theta)$$

The representation of a token is inextricably linked to the specific instantiation of its neighbors. In BERT, the vector for "bank" in sentence 1 is mathematically different from the vector for "bank" in sentence 2.

2 BERT Architecture: Theoretical Deep Dive

BERT is a multi-layer bidirectional Transformer Encoder. To understand it, we must dissect the input representation and the attention mechanism.

2.1 Input Representation

BERT does not just take words. Its input representation is a sum of three distinct embeddings:

$$\mathbf{E}_{\text{input}} = \mathbf{E}_{\text{token}} + \mathbf{E}_{\text{segment}} + \mathbf{E}_{\text{position}} \tag{1}$$

2.1.1 1. Token Embeddings (WordPiece)

BERT uses **WordPiece** tokenization. Unlike standard word splitting, WordPiece breaks unknown words into subword units.

- "playing" → ["play", "ing"]
- "investment" → ["invest", "ment"]

This solves the **Out-of-Vocabulary (OOV)** problem. If the model sees a new complex financial term like "Cryptoleverage", it can break it down into "Crypto", "lever", "age"—parts it already understands.

2.1.2 2. Positional Embeddings

Since the Transformer reads the entire sentence at once (in parallel), it has no inherent sense of order. It doesn't know that "Man bites dog" is different from "Dog bites man." To fix this, BERT learns a specific vector for "Position 1", "Position 2", etc., and adds this to the word embedding.

2.2 The Transformer Encoder Block

Each layer in BERT applies the **Self-Attention** mechanism.

Theoretical Detail: Mathematical Detail: Scaled Dot-Product Attention

The core engine of BERT is the attention function:

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

The Components:

- **Query (Q):** What the current token is looking for.
- **Key (K):** What every other token identifies as.
- **Value (V):** The actual content of the token.

The Intuition: Think of a database. You have a query (Q). You match it against keys (K) in the database. When you find a match, you retrieve the value (V). In BERT, every word queries every other word. "Bank" queries the sentence. "River" answers (high dot product score). "Bank" then updates its meaning to be more "geological".

2.3 GELU: The Activation Function

BERT uses the **Gaussian Error Linear Unit (GELU)** instead of the standard ReLU.

$$\text{GELU}(x) = x\Phi(x)$$

where $\Phi(x)$ is the cumulative distribution function of the Gaussian distribution. **Why?** ReLU is deterministic ($x > 0$?). GELU introduces a stochastic aspect, smoothing the decision boundary. This curvature helps optimization in deep Transformers where the loss landscape is highly non-convex.

3 Pre-training Objectives

BERT is not trained to predict the next word (like GPT). It is trained as a **Denoising Autoencoder**.

3.1 Task 1: Masked Language Modeling (MLM)

BERT randomly hides 15% of the input tokens and attempts to reconstruct them.

The Manifold Hypothesis: Why does masking work? Natural language lies on a "manifold" (a structured surface) within the high-dimensional space of random character combinations. By damaging the input (masking) and forcing the model to project it back onto the manifold (predicting the correct word), the model learns the internal laws of syntax and semantics.

Insight: The 80-10-10 Rule

The 15% masking strategy is subtle:

- **80% [MASK]:** Replace word with [MASK]. (Core learning)
- **10% Random Word:** Replace "Market" with "Apple". (Teaches the model to trust context over the input symbol itself).
- **10% Original Word:** Leave "Market" as "Market". (Prevents the model from thinking that *every* input is wrong).

3.2 Task 2: Next Sentence Prediction (NSP)

- **Input:** [CLS] Sentence A [SEP] Sentence B
- **Objective:** Binary Classification. Does B logically follow A?
- **Relevance:** This is crucial for tasks like Question Answering or Natural Language Inference, where the relationship between two distinct segments of text matters.

4 FinBERT: Domain Adaptation

While BERT achieves state-of-the-art results on general English (Wikipedia), it struggles with the specific nuances of financial language.

4.1 The Domain Shift Problem

Financial documents (10-Ks, Earnings Calls) have a distribution of language that is statistically distinct from Wikipedia.

Real-Life Example: Semantic Drift in Finance

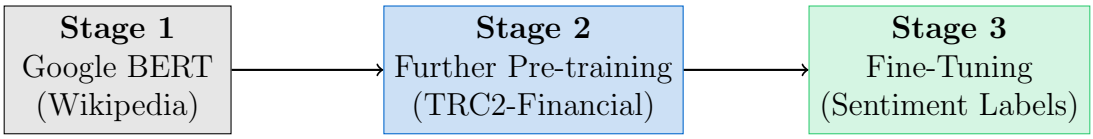
Common words often possess entirely different sentiments or meanings in a financial context:

- **"Liability":**
  - General English: "He is a liability to the team" (Negative).
  - Finance: "Current Liabilities" (Neutral Accounting Term).
- **"Vice":**
  - General English: Immoral behavior / Sin.
  - Finance: "Vice President" (Neutral/Positive title).
- **"Share":**
  - General English: To distribute or give.
  - Finance: A unit of equity ownership.

Standard BERT tends to misclassify financial "negative" words (like "cost", "loss", "liability") based on their general usage rather than their financial neutrality.

4.2 FinBERT Training Pipeline

FinBERT is a **BERT-Base** model that has undergone a specific training pipeline known as **Transfer Learning with Domain Adaptation**.



4.2.1 Stage 2: The "Magic" Step (Further Pre-training)

This is the critical step. We take the Google BERT weights and continue the **Masked Language Modeling (MLM)** task, but this time, we only feed it Financial Text.

The Dataset: TRC2-Financial

- **Source:** Thomson Reuters Corpus (Volume 2).
- **Content:** Financial news articles (2008-2010).
- **Size:** 46,000 articles containing 29 million words.

Theoretical Detail: Mathematics of Adaptation

Mathematically, general pre-training places the model parameters θ in a region of the loss landscape optimal for General English (P_{gen}). Further pre-training moves θ to a new region optimal for Financial English (P_{fin}).

$$\theta_{fin} = \text{argmin}_\theta \mathbb{E}_{x \sim P_{fin}}[\mathcal{L}_{\text{MLM}}(x, \theta)]$$

Crucially, because we initialize from θ_{gen}, the optimization path requires fewer steps than training from scratch.

4.2.2 Stage 3: Supervised Fine-Tuning

Once the model "speaks finance" (Stage 2), we teach it to perform Sentiment Analysis. We attach a classification head on top of the '[CLS]' token and train on the **Financial PhraseBank** dataset (Malo et al., 2014), which contains expert-annotated sentences (Positive, Negative, Neutral).

4.3 Performance Comparison

How much does this elaborate process actually help?

Model	**Accuracy**	**F1 Score**
Loughran-McDonald (Dictionary)	60-65%	0.55
LSTM (GloVe embeddings)	75-78%	0.72
Standard BERT (bert-base)	80-85%	0.83
FinBERT	**97%**	**0.96**

5 Implementation Guide

5.1 HuggingFace Transformers Implementation

We use the ‘ProsusAI/finbert‘ model, which is the industry standard open-source implementation.

Code Implementation

```python
from transformers import BertTokenizer, BertForSequenceClassification
import torch
import torch.nn.functional as F

# 1. Load Pre-trained FinBERT
tokenizer = BertTokenizer.from_pretrained('ProsusAI/finbert')
model = BertForSequenceClassification.from_pretrained('ProsusAI/finbert'
    , use_safetensors=True)

# 2. Financial Text Input
text = "The company reported a 20% decrease in revenue, missing
    expectations."

# 3. Tokenization (Note the truncation)
inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=
    True)

# 4. Inference
with torch.no_grad():
    outputs = model(**inputs)
    logits = outputs.logits

# 5. Probabilities via Softmax
probabilities = F.softmax(logits, dim=1)
# Output: [Positive, Negative, Neutral]
print(probabilities)
```

5.2 Interpreting the Output

FinBERT (ProsusAI) outputs 3 logits corresponding to labels:

1. **Positive:** Growth, profit increase, exceeding estimates, upgrades.
2. **Negative:** Loss, revenue decline, missed estimates, lawsuits.
3. **Neutral:** Factual statements without sentiment (e.g., ”The company is headquartered in London” or ”The meeting is at 5 PM”).

6 Advanced Topics Challenges

Industry-Standard Challenges

In production environments, simply running 'model(text)' is rarely enough. Here are the advanced challenges you will face.

6.1 Challenge 1: Catastrophic Forgetting

When we train on Reuters data (Stage 2), the model might "forget" knowledge about non-financial topics (e.g., biology or literature) acquired during Stage 1.

Insight: In FinBERT, we arguably *want* catastrophic forgetting. We want the model to forget that "depression" is a mental health state and learn that "depression" is an economic state. The shift in vector space is a feature, not a bug.

6.2 Challenge 2: The 512-Token Limit

BERT has a hard limit of 512 tokens. Financial 10-K reports are often 50,000+ words.

Strategies:

- **Truncation:** Keep the first 512 tokens (Header/Intro). *Risk: Missing info at the end.*
- **Hierarchical approach:** Chunk the document into 512-token segments. Run FinBERT on each chunk. Average the sentiment scores.
- **Longformer/BigBird:** Use newer architectures designed for long contexts (Attention $O(N)$ instead of $O(N^2)$).

6.3 Challenge 3: Slanted Triangular Learning Rates

During the fine-tuning of FinBERT, researchers often employ **Slanted Triangular Learning Rates** (Howard & Ruder, 2018).

Theoretical Detail: STLR Schedule

Instead of a constant learning rate, we use a schedule that:

1. **Linearly increases** (warm-up) the learning rate to a max value η_{max} quickly (e.g., first 10% of steps). This allows the gradients to align.
2. **Linearly decays** the learning rate to η_{min} over the remaining 90% of steps. This allows fine-grained convergence.

7 Week 2 Completion Checklist

Essential Tasks

Theoretical Mastery:

- ✓ Explain why Static Embeddings fail on the word "Bank" and the importance of BERT architecture.
- ✓ Draw the 3 components of the BERT input embedding.
- ✓ Explain the 80-10-10 Masking Rule.

FinBERT Specifics:

- ✓ Understand the 3-stage training pipeline.
- ✓ Define "Domain Adaptation" in the context of NLP.
- ✓ Explain why we use TRC2-Financial for pre-training and Financial PhraseBank for fine-tuning.

Implementation:

- ✓ Load 'ProsusAI/finbert' in a Jupyter Notebook.
- ✓ Write a function that takes a long document, chunks it into 512-token segments, and returns the average sentiment.