



Neural Networks and Deep Learning

Lecture 7: Attention and transformers



Prof. dr. Daniel Traian PELE

Bucharest University of Economic Studies (ASE), Department of Statistics and Econometrics
Romanian Academy, Institute for Economic Forecasting (IPE)
IDA Institute Digital Assets ▪ AI4EFin ▪ MSCA Digital Finance

Learning objectives

- Explain why attention replaced recurrence for text: one step between any two words, computed in parallel.
- Write scaled dot-product attention, multi-head attention, positional encoding and the transformer block with every symbol defined.
- Compare, on real finance tweets, a small transformer trained from scratch, a pretrained Bidirectional Encoder Representations from Transformers (BERT) model for finance (FinBERT) and a term frequency–inverse document frequency (TF-IDF) benchmark.
- Read attention maps with care, and decide when pretraining, fine-tuning or a simple model is the right business choice.

Abbreviations

ASE	Bucharest University of Economic Studies
BERT	Bidirectional Encoder Representations from Transformers
CNN	convolutional neural network
ECB	European Central Bank
ES	expected shortfall
FFN	position-wise feed-forward network
FinBERT	BERT further trained on financial text
GELU	Gaussian error linear unit
GPT	Generative Pre-trained Transformer
IPE	Institute for Economic Forecasting, Romanian Academy
LLM	large language model
LN	layer normalisation
LSTM	long short-term memory network
MHA	multi-head attention
MLM	masked language modelling
MPS	Metal Performance Shaders (Apple graphics processor back end)
NLP	natural language processing
ReLU	rectified linear unit
RNN	recurrent neural network
TF-IDF	term frequency–inverse document frequency
VaR	Value-at-Risk

Outline

- Motivation for the attention mechanism
- The attention mechanism
- The transformer architecture
- Case study: sentiment of finance tweets
- Transformers in practice
- Wrap-up

Financial news as input for models

- **In one sentence:** Thousands of headlines and posts arrive every hour; a desk needs their tone in seconds, consistently, in the same units.

$$\hat{\mathbf{p}}(x_{1:n}) = \text{softmax}\left(W_o\,\mathbf{h}(x_{1:n}) + \mathbf{b}_o\right) \in \Delta^3$$

- **where**
  - $x_{1:n}$ = the tokens of one text
  - $\mathbf{h}$ = a fixed-size representation of the text
  - Δ^3 = probabilities of bearish, bullish, neutral
  - The whole lecture is about how to compute $\mathbf{h}$.

- **Our data**
  - 9 543 labelled finance tweets for training, 2 388 for the test
  - Test: 14.5 % bearish, 19.9 % bullish, 65.6 % neutral

- **Evidence**
  - Media pessimism predicts next-day returns and volume (Tetlock, 2007)

- **In practice:** a sentiment model is a measurement instrument: it must be accurate on the rare bearish news, not only on average.

(Tetlock, 2007; Malo et al., 2014)

Limits of recurrent networks for long texts

- **In one sentence:** A recurrent network squeezes everything read so far into one state vector, updated one token at a time.

$$\mathbf{h}_t = \phi\big(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + \mathbf{b}\big), \qquad \frac{\partial \mathbf{h}_n}{\partial \mathbf{h}_1} = \prod_{t=2}^{n} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}$$

- **where**
  - $\mathbf{x}_t$ = embedding of token t
  - $\mathbf{h}_t$ = state
  - ϕ = tanh
  - The product of $n-1$ Jacobians shrinks or explodes with the distance between words.

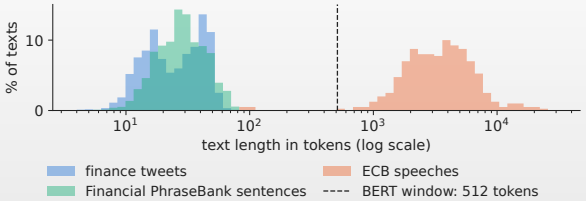
- **Two costs**
  - Path between words i and j: $|i-j|$ steps, so long links fade
  - n sequential steps: no parallelism over the sentence

- **Gates help, not cure**
  - Long short-term memory (LSTM) of Lecture 5 eases but keeps the sequential path

- **In practice:** in "profit fell, but guidance was raised", the verdict depends on a word far from the subject.

(Bahdanau, Cho & Bengio, 2015; Bishop & Bishop, 2024, sec. 12.1)

Length of tweets and central-bank speeches in tokens



Source: Twitter financial news (zeroshot); Malo et al. (2014); European Central Bank speeches; tokenizer of the ProsusAI model.
Data: finance tweets; Financial PhraseBank; European Central Bank speeches 8 Jan 2022 to 30 Oct 2026.

- **Median length in tokens**: tweets 25, Financial PhraseBank sentences 28, European Central Bank (ECB) speeches 3 338.
- **Context limit**: 98.8 % of the 400 speeches exceed 512 tokens
  - Only 0.19 % of the tweets exceed 64.

- **In practice:** match the model to the text length: a tweet fits any transformer, a speech needs chunking or a long-context model.

NN_L7_context_lengths

Outline

- Motivation for the attention mechanism
- The attention mechanism
- The transformer architecture
- Case study: sentiment of finance tweets
- Transformers in practice
- Wrap-up

Attention is a soft dictionary lookup

- **In one sentence:** Each word asks a question (query), every word offers a label (key) and a content (value); the answer is the values weighted by how well keys match the query.

$$\mathbf{y}_i = \sum_{j=1}^{n} a_{ij}\,\mathbf{v}_j, \qquad a_{ij} = \frac{\exp\left(\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d_k}\right)}{\sum_{l=1}^{n} \exp\left(\mathbf{q}_i^\top \mathbf{k}_l / \sqrt{d_k}\right)}$$

$$\mathbf{q}_i = W_Q^\top \mathbf{x}_i, \quad \mathbf{k}_j = W_K^\top \mathbf{x}_j, \quad \mathbf{v}_j = W_V^\top \mathbf{x}_j$$

- **where**
  - $\mathbf{x}_i \in \mathbb{R}^d$ = token representation
  - $W_Q, W_K \in \mathbb{R}^{d \times d_k}$, $W_V \in \mathbb{R}^{d \times d_v}$ = learned projections
  - $a_{ij} \geq 0$, $\sum_j a_{ij} = 1$: a probability over the words.

- **Hard lookup**
  - A dictionary returns the value of the one matching key
- **Soft lookup**
  - Attention returns a blend of all values: differentiable, so W_Q, W_K, W_V are learned
  - One step between any two words, whatever their distance

- **In practice:** each output word is a context-aware mixture: "guidance" can borrow meaning from "raised" ten words later.

(Vaswani et al., 2017; Bishop & Bishop, 2024, sec. 12.1)

Experiment yourself: notebook §2

Attention in matrix form for all words at once

⊡ **In one sentence:** Stacking queries, keys and values as matrices turns attention into two matrix products and a row-wise softmax.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + \mathbf{M}\right)V$$
$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

⊡ **where**

- $X \in \mathbb{R}^{n \times d}$ = the n token representations
- $QK^\top \in \mathbb{R}^{n \times n}$ = all pairwise scores
- $\mathbf{M}$ = mask, 0 for allowed pairs and $-\infty$ for padding (or for future tokens in a decoder).

⊡ **Cost per layer**

- $O(n^2 d)$ operations and $O(n^2)$ memory
- A median speech costs about 17 828 times a median tweet

⊡ **Parallel**

- No loop over positions: graphics processors multiply the whole sentence at once

⊡ **In practice:** the n^2 term is why context windows are limited and long reports must be cut into pieces.

(Vaswani et al., 2017)

Experiment yourself: notebook §2

Scaling by $\sqrt{d_k}$ and saturation of the softmax

⊡ **In one sentence:** Dot products of long random vectors grow with their length; without scaling the softmax becomes one-hot and its gradient vanishes.

$$q_l, k_l \overset{\text{i.i.d.}}{\sim} (0,1) \;\Rightarrow\; \mathbb{E}[\mathbf{q}^\top \mathbf{k}] = 0, \quad \mathrm{Var}(\mathbf{q}^\top \mathbf{k}) = \sum_{l=1}^{d_k} \mathrm{Var}(q_l k_l) = d_k$$

$$\mathrm{Var}\left(\mathbf{q}^\top \mathbf{k} / \sqrt{d_k}\right) = 1 \quad \text{for every } d_k$$

⊡ **where**

- d_k = dimension of queries and keys (64 per head in BERT)
- With unit-variance scores the softmax spreads its mass and its Jacobian $\mathrm{diag}(\mathbf{a}) - \mathbf{a}\mathbf{a}^\top$ stays away from zero.

⊡ **Without scaling**

- With $d_k = 64$ the scores have standard deviation 8, so one word takes almost all the weight

⊡ **Temperature view**

- $\sqrt{d_k}$ is a fixed temperature that keeps attention soft at initialisation

⊡ **In practice:** a design constant, not a tuning knob: it is what lets deep stacks of attention train at all.

(Vaswani et al., 2017)

Multi-head attention: several relations at once

- **In one sentence:** One head averages one kind of relation; several heads with smaller projections capture syntax, negation and entities in parallel.

$$\mathrm{MHA}(X) = \left[\mathbf{H}_1 \,\|\, \cdots \,\|\, \mathbf{H}_h\right] W_O$$
$$\mathbf{H}_m = \mathrm{Attention}(XW_Q^m, XW_K^m, XW_V^m)$$

- **where**
  - h = number of heads
  - $W_Q^m, W_K^m, W_V^m \in \mathbb{R}^{d \times d/h}$
  - $\|$ = concatenation
  - $W_O \in \mathbb{R}^{d \times d}$
  - Parameters: $4d^2$ (+ biases), the same as one big head.

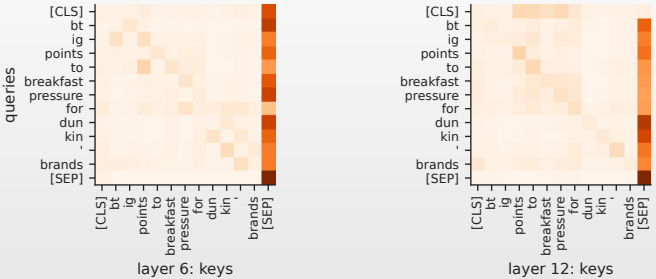
- **BERT-base**
  - 12 layers $\times$ 12 heads, $d = 768$, $d/h = 64$
- **Our small transformer**
  - 2 layers $\times$ 4 heads, $d = 64$, $d/h = 16$

- **In practice:** heads are cheap specialists: same cost as one head, more kinds of context per layer.

(Vaswani et al., 2017; Clark et al., 2019)

Experiment yourself: notebook §2

What FinBERT attends to in a real bearish tweet



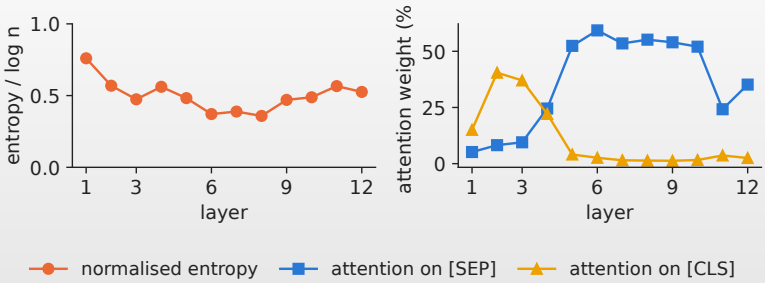
Source: Twitter financial news datasets (Hugging Face, zeroshot). Data: one test tweet; pretrained FinBERT, attention averaged over 12 heads.

- **Tweet** (test set, label bearish): "BTIG points to breakfast pressure for Dunkin' Brands"
  - Pretrained FinBERT gives it only 7.3 % probability of negative: the bearish tone is implicit.
- **Layer 6**: the [CLS] token puts 5.7 % of its attention on "points"
  - In layer 12 most weight flows to the separator tokens.

(own computation; mean over the 12 heads)

NN_L7_attention_example

Across layers, attention becomes concentrated on separator tokens



Source: Twitter financial news datasets (Hugging Face, zeroshot). Data: 500 test tweets; pretrained FinBERT.

- **Entropy** of the attention rows, divided by $\log n$: 0.76 in layer 1, lowest 0.36 in layer 8 (1 = uniform).
- **[SEP] as a sink**: up to 59.3 % of all attention (layer 6) goes to the separator: a "no-op" a head uses when it has nothing to add.

(own computation, 500 test tweets; Clark et al., 2019)

NN_L7_attention_entropy

Attention weights as explanations: limitations

- **In one sentence:** Where a head looks is not the same as why the model decides: values, later layers and the classifier all transform what attention selects.

$$\mathbf{h}_i^{(l+1)} = \mathbf{h}_i^{(l)} + \sum_{m=1}^{h} \sum_{j} a_{ij}^{(l,m)} \, W_O^{m\top} W_V^{m\top} \mathbf{h}_j^{(l)} + \dots$$

- **where**
  - The contribution of word j is a_{ij} times a value vector whose norm can be large or small
  - The residual stream $\mathbf{h}_i^{(l)}$ carries information that attention never touches.

- **Evidence**
  - Different attention maps can give the same prediction (Jain & Wallace, 2019)
  - Heavy attention on [SEP] carries almost no information (Clark et al., 2019)

- **Better tools**
  - Gradient-based attributions and occlusion tests (Lecture 11)

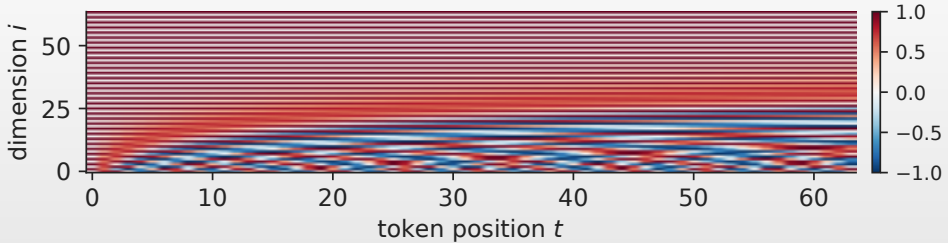
- **In practice:** never show an attention map to a regulator as "the reason" for a decision.

(Jain & Wallace, 2019; Clark et al., 2019)

Outline

- Motivation for the attention mechanism
- The attention mechanism
- The transformer architecture
- Case study: sentiment of finance tweets
- Transformers in practice
- Wrap-up

Sinusoidal positional encoding



Source: Vaswani et al. (2017), sinusoidal formula. Data: mechanism illustration, $d = 64$, 64 positions.

- **Formula**: $p_{t,2i} = \sin(t/10000^{2i/d})$, $p_{t,2i+1} = \cos(t/10000^{2i/d})$, added to the token embedding
  - Low dimensions change fast, high ones slowly.
- **Why**: self-attention is permutation-equivariant
  - Without positions "rates cut, stocks up" equals "stocks cut, rates up".

SIMULATED DATA (teaching illustration) (Vaswani et al., 2017) NN_L7_positional_encoding

A transformer block: attention, feed-forward, residuals, normalisation

⊡ **In one sentence:** Attention mixes information across words; a small two-layer network then transforms each word on its own; residual links and layer normalisation keep training stable.

$$Z = X + \mathrm{MHA}\big(\mathrm{LN}(X)\big), \qquad X' = Z + \mathrm{FFN}\big(\mathrm{LN}(Z)\big)$$

$$\mathrm{FFN}(\mathbf{z}) = W_2\ \mathrm{GELU}(W_1\mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2, \quad \mathrm{LN}(\mathbf{z}) = \gamma \odot \frac{\mathbf{z} - \mu_{\mathbf{z}}}{\sigma_{\mathbf{z}}} + \beta$$

⊡ **where**

- Pre-norm variant
- Position-wise feed-forward network (FFN), $W_1 \in \mathbb{R}^{4d \times d}$
- Gaussian error linear unit (GELU) or rectified linear unit (ReLU)
- Layer normalisation (LN) over the d features of one token
- Parameters per block $\approx 12d^2$.

⊡ **Residuals**

- Gradients reach early layers directly (Lecture 4)

⊡ **Our block**

- $d = 64$, FFN width 128, dropout 0.1; 2 blocks: 67 139 parameters outside the embeddings

⊡ **In practice:** most parameters of a transformer sit in the feed-forward layers, not in attention.

(Vaswani et al., 2017; Ba, Kiros & Hinton, 2016)

Experiment yourself: notebook §3

Encoders and decoders: two pretraining objectives

- **In one sentence:** The same block reads or writes, depending on the mask and the pretraining task.

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log p_\theta\left(x_i \mid x_{\setminus \mathcal{M}}\right)$$

$$\mathcal{L}_{\text{LM}} = -\sum_{t=1}^{n} \log p_\theta\left(x_t \mid x_{<t}\right)$$

- **where**
  - $\mathcal{M} = 15\,\%$ of positions, masked at random: masked language modelling (MLM)
  - Second line: next-token prediction, causal mask $M_{ij} = -\infty$ for $j > i$.

- **Encoder-only**
  - Bidirectional Encoder Representations from Transformers (BERT): classification, extraction
- **Decoder-only**
  - Generative Pre-trained Transformer (GPT): text generation, today's large language models (LLM)

- **In practice:** for labelling texts use an encoder
  - For writing reports use a decoder (Lecture 12).

(Devlin et al., 2019; Radford et al., 2018)

Subword tokenisation of text

- **In one sentence:** A fixed vocabulary of word pieces covers any word, including tickers and new company names, by splitting rare words into frequent pieces.

$$x_{1:n} = \big[\texttt{[CLS]}, w_1, \ldots, w_{n-2}, \texttt{[SEP]}\big], \qquad \mathbf{x}_t = E[x_t] + \mathbf{p}_t + \mathbf{s}_t$$

- **where**
  - $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ = embedding table
  - $\mathbf{p}_t$ = position, $\mathbf{s}_t$ = segment embedding
  - CLS summarises the text, [SEP] closes it
  - WordPiece vocabulary of FinBERT: $|\mathcal{V}| = 30\,522$.

- **Our tweets**
  - Median 25 tokens, 99th percentile 57, maximum 82
  - We cut at 64 tokens: 0.19 % of the tweets lose a few tokens

- **Embeddings dominate small models**
  - Small transformer: 2 020 547 parameters, of which 67 139 outside the table

- **In practice:** check how a tokenizer splits your tickers and product names before trusting a model on them.

(Wu et al., 2016; Devlin et al., 2019)

Experiment yourself: notebook §1

Attention, recurrence and convolution compared

Layer type	Cost per layer	Sequential steps	Longest path
Self-attention	$O(n^2 d)$	$O(1)$	$O(1)$
Recurrent (RNN, LSTM)	$O(nd^2)$	$O(n)$	$O(n)$
Convolution, kernel k	$O(knd^2)$	$O(1)$	$O(\log_k n)$

- **Recurrent neural network (RNN)**: cheap per step, slow per sentence, long paths
  - **convolutional neural network (CNN)**: parallel, but needs depth to link distant words.
- **Self-attention**: constant path and full parallelism, paid with n^2 memory
  - For tweets ($n \approx 25$) the quadratic term is negligible.

(Vaswani et al., 2017, Table 1)

Outline

- Motivation for the attention mechanism
- The attention mechanism
- The transformer architecture
- Case study: sentiment of finance tweets
- Transformers in practice
- Wrap-up

Seven models evaluated on a common test set

Model	What is trained
TF-IDF + logistic	logistic regression on word 1–2-grams (17 119 features), C chosen on validation
Embedding bag	mean of learned token embeddings, no attention
Small transformer	2 blocks, 4 heads, $d = 64$, from scratch; also without positions
FinBERT zero-shot	BERT further trained on financial text (FinBERT, Araci, 2019), used as is
FinBERT frozen + logistic	logistic regression on the mean last-layer vector (768)
FinBERT fine-tuned	all 109.5 million weights, 3 epochs

- **Split**: published training tweets = 8 588 training + 955 validation (10 %, stratified)
  - The published validation set = our test (2 388), used once.
- **Why not Financial PhraseBank**: FinBERT was fine-tuned on it, so testing there would leak
  - Tweets are new to every model.

(Araci, 2019; Malo et al., 2014)

Experiment yourself: notebook §5

Our small transformer, trained from scratch

- **In one sentence:** Two blocks on top of token and position embeddings, mean-pooled over the real tokens, then a linear layer to the three classes.

$$\mathbf{h} = \frac{1}{n}\sum_{t=1}^{n} \mathbf{x}_t^{(2)}, \qquad \mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} \log \operatorname{softmax}(W\mathbf{h}_i + \mathbf{b})_{y_i}$$

- **where**
  - $\mathbf{x}_t^{(2)}$ = output of the second block for token t
  - w_c = class weight, inversely proportional to the class frequency
  - Adam with decoupled weight decay (AdamW), learning rate $5 \cdot 10^{-4}$, batch 64, early stopping on validation macro-F1.

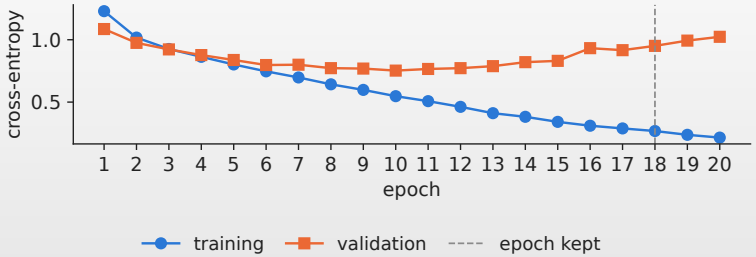
- **Size**
  - 2 020 547 parameters, mostly the embedding table
- **Cost**
  - 44 s per seed; median 20 epochs before stopping
- **Macro-F1**
  - Mean of the three per-class F1 scores: bearish counts as much as neutral

- **In practice:** macro-F1 is the right yardstick when the rare class (bearish news) is the one that costs money.

(Loshchilov & Hutter, 2019)

Experiment yourself: notebook §5

The small transformer overfits: the validation loss turns up after a few epochs

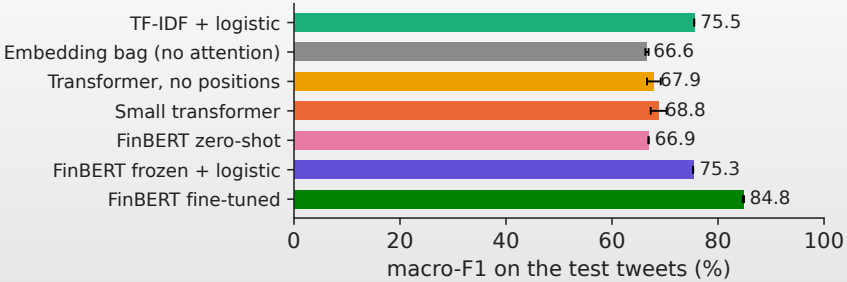


Source: Twitter financial news datasets (Hugging Face, zeroshot). Data: own estimates; small transformer, seed 1.

- **Seed 1**: the validation loss is lowest at epoch 10 (0.753) and rises to 1.024 by epoch 20
  - The macro-F1 used for stopping still peaks at epoch 18.
- **Data-hungry**: 8 588 tweets are few for 2 020 547 parameters learned from nothing
  - The model grows overconfident before it stops improving.

NN_L7_tiny_learning_curve

Comparison of the models: macro-F1 on the test set



Source: Twitter financial news datasets (Hugging Face, zeroshot). Data: own estimates; 2,388 test tweets (published validation split).

- **Macro-F1 on 2 388 test tweets**: FinBERT fine-tuned 84.8 %, TF-IDF 75.5 %, small transformer 68.8 % (sd 1.5, 3 seeds).
- **FinBERT without training on tweets**: 66.9 %
  - Frozen FinBERT + logistic: 75.3 %.

NN_L7_model_comparison

The bearish class is where models differ most

Model	Macro-F1	Accuracy	Bearish recall	Bearish precision	Runs
TF-IDF + logistic	75.5	81.5	70.3	63.5	1
Embedding bag	66.6	74.3	62.0	46.6	3
Transformer, no positions	67.9	75.4	61.8	48.4	3
Small transformer	68.8	76.7	59.7	51.1	3
FinBERT zero-shot	66.9	72.6	76.1	49.4	1
FinBERT frozen + logistic	75.3	80.1	78.7	58.6	1
FinBERT fine-tuned	84.8	88.2	82.9	77.9	3

- **All numbers in %**, test tweets
  - Green = best, red = worst in the column
  - Means over the runs.
- **Bearish news**: fine-tuned FinBERT finds 82.9 % of them with 77.9 % precision
  - TF-IDF 70.3 % and 63.5 %.

In practice: report per-class recall and precision: an average can hide a model that misses the news that matter.

(own computation)

Experiment yourself: notebook §5

Paired bootstrap test of the differences on the test tweets

- **In one sentence:** Resampling the test tweets, with the same draw for both models, shows how much a gap in macro-F1 could move by chance.

$$\Delta^{*b} = F_1^A(\mathcal{T}^{*b}) - F_1^B(\mathcal{T}^{*b}), \quad b = 1, \ldots, 1000$$
$$\mathrm{CI}_{95\%} = [q_{0.025}, q_{0.975}]$$

- **where**
  - $\mathcal{T}^{*b}$ = test tweets drawn with replacement
  - F_1 = macro-F1 averaged over the seeds of the model
  - Differences in percentage points.

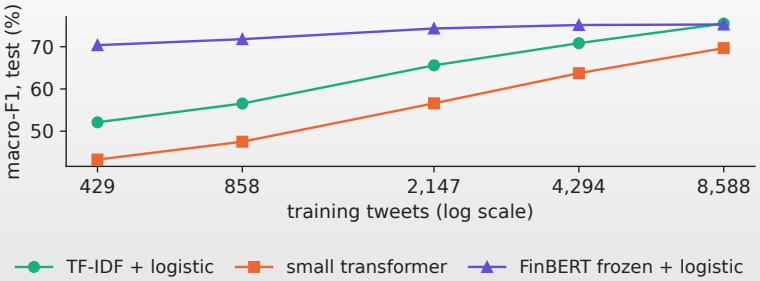
- **FinBERT fine-tuned − TF-IDF**
  - $+9.3$ $[+7.3; +11.3]$
- **Small transformer − TF-IDF**
  - -6.6 $[-8.3; -4.9]$
- **With − without positions**
  - $+0.9$ $[+0.1; +1.7]$
- **Frozen FinBERT − TF-IDF**
  - -0.2 $[-2.5; +2.0]$

- **In practice:** an interval that excludes zero is the minimum evidence before replacing a production model.

(Efron & Tibshirani, 1993)

Experiment yourself: notebook §5

With little labelled data, pretrained representations matter most



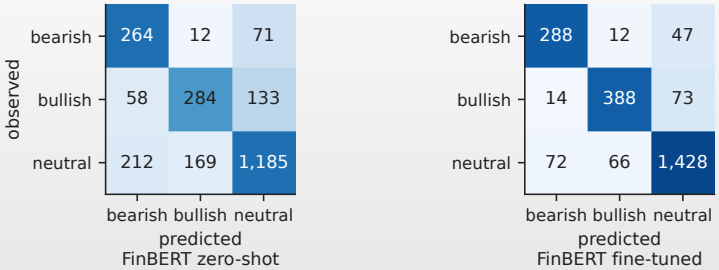
Source: Twitter financial news datasets (Hugging Face, zeroshot). Data: own estimates; 2,388 test tweets (published validation split).

- **With 429 training tweets**: frozen FinBERT + logistic 70.4 %, TF-IDF 52.1 %, small transformer 43.3 % macro-F1.
- **Pretraining = data you did not label**: billions of words already shaped the FinBERT vectors
  - A model from scratch must learn language and task at once.

(own computation, seed 1)

NN_L7_data_scaling

Zero-shot errors corrected by fine-tuning



Source: Twitter financial news datasets (Hugging Face, zeroshot). Data: own estimates; 2,388 test tweets (published validation split).

- **Zero-shot**: FinBERT was trained on news sentences, not tweets
  - It finds 264 of the 347 bearish tweets
  - After fine-tuning: 288.
- **Remaining errors**: 47 bearish and 73 bullish tweets read as neutral
  - Only 12 bearish read as bullish.

(own computation, fine-tuned model of seed 1)

NN_L7_confusion

Fine-tuning: pretrained initial weights and small learning rates

- **In one sentence:** All weights move, but only a little: the language knowledge stays, the decision layer adapts to tweets.

$$\theta_0 = \theta_{\text{FinBERT}}, \qquad \theta_{s+1} = \theta_s - \eta_s \widehat{\nabla \mathcal{L}}(\theta_s), \quad \eta_s = \eta_0 \left(1 - \frac{s}{S}\right)$$

- **where**
  - $\eta_0 = 2 \cdot 10^{-5}$, linear decay over $S = 3$ epochs of batches of 32
  - AdamW with weight decay 0.01
  - The epoch with the best validation macro-F1 is kept.

- **Gain**
  - +17.9 points over zero-shot, +9.3 over TF-IDF
- **Cost**
  - 109.5 million weights, 5.2 minutes per seed on a laptop graphics processor
- **Risk**
  - A large learning rate erases pretraining (catastrophic forgetting)

- **In practice:** fine-tuning a pretrained encoder is today's default for any labelled text task with a few thousand examples.

(Devlin et al., 2019; Araci, 2019)

Experiment yourself: notebook §5

Role of word order in tweet classification

- **Ablation**
  - Same small transformer with and without positional encoding, 3 seeds each
  - Macro-F1 68.8 % with positions, 67.9 % without: +0.9 points
- **Bootstrap**
  - 95 % interval of the gain: [+0.1; +1.7] points
- **Reading**
  - Short, formulaic tweets are mostly bags of signal words (“cut”, “beats”, “downgrade”)
  - The embedding bag, with no attention, is close: 66.6 %
- **When order matters**
  - Negation and contrast (“not”, “but”), long documents, numbers attached to the right entity

- **In practice:** test the simple version of your model first: on short texts much of the transformer’s power is unused.

(own computation)

Experiment yourself: notebook §5

Outline

- Motivation for the attention mechanism
- The attention mechanism
- The transformer architecture
- Case study: sentiment of finance tweets
- Transformers in practice
- Wrap-up

Sentiment scores as a business signal

- **Pipeline**
  - Stream of news and posts → tokenizer → fine-tuned encoder → class probabilities
  - Aggregate by firm and day: share of bearish minus bullish, weighted by reach
- **Uses**
  - Risk monitoring of a credit portfolio, early warning for issuers, an input to Value-at-Risk (VaR) overlays
- **Pitfalls**
  - Look-ahead: use only texts time-stamped before the decision
  - Domain shift: a model of news sentences misreads tweets (zero-shot gap)
  - Label noise and sarcasm; drift when the market vocabulary changes

- **In practice:** a sentiment score is an input to a model, not a trading decision: backtest it before use.

(Tetlock, 2007)

Long documents: central-bank speeches and annual reports

- **Our speeches**
  - 400 speeches of the ECB, 8 Jan 2022 to 30 Oct 2026
  - Median 3 338 tokens, longest 22 646
- **Quadratic cost**
  - Attention on a median speech costs about 17 828 times that on a median tweet
- **Options**
  - Chunk into windows of 512 and pool the chunk scores
  - Hierarchical models: sentence vectors, then attention over sentences
  - Long-context models: sparse or memory-efficient attention

- **In practice:** for a monetary-policy desk, chunking plus a well-tested encoder usually beats an untested long-context model.

(Vaswani et al., 2017)

Experiment yourself: notebook §1

Transformers for data other than text

- **Time series**
  - Patches of a series as tokens; pretrained forecasting models (Lecture 13)
- **Tables and transactions**
  - Each column or each past payment as a token; attention across them
- **Graphs**
  - Attention restricted to the edges = graph attention network (Lecture 9)
- **Images and documents**
  - Image patches as tokens; vision-language models read charts and reports (Lecture 14)

- **In practice:** whatever the data, the question is the same: what are the tokens, and which pairs may attend to each other?

(Bishop & Bishop, 2024, ch. 12)

Outline

- Motivation for the attention mechanism
- The attention mechanism
- The transformer architecture
- Case study: sentiment of finance tweets
- Transformers in practice
- Wrap-up

Key takeaways

- **Attention** = softmax of scaled query-key scores times values
  - One step between any two words, $O(n^2)$ cost
  - $\sqrt{d_k}$ keeps the softmax trainable.
- **Transformer block** = multi-head attention + feed-forward, each with a residual link and layer normalisation
  - Positions are added, because attention ignores order.
- **On finance tweets** pretraining decides: fine-tuned FinBERT 84.8 % macro-F1, TF-IDF 75.5 %, a transformer from scratch 68.8 %.
- **Attention maps** are clues, not explanations
  - Check per-class errors and bootstrap intervals before trusting a model.

Research from our group: transformers as risk forecasters

- **Large language models (LLM) as tail-risk forecasters** (Pele, Bolovăneanu, Lin, Ren, Ginavar, Spilak, Andrei, Toma, Lessmann & Härdle, 2026): uses decoder transformers, prompted with return histories, to forecast Value-at-Risk and the expected shortfall (ES) — LLM-VaR and LLM-ES — and evaluates them with standard risk backtests.
- **Foundation models for tail risk** (Jheng, Tak, Găman, Mazurencu-Marinescu-Pele & Pele, 2026): uses pretrained transformer forecasting models, without retraining and with conformal calibration, to forecast VaR and ES in Central and Eastern European markets.
- **Link with this lecture**: the same idea as FinBERT, a model pretrained on large data and used with little or no task-specific training, applied to risk numbers instead of words.

Findings as stated in the papers; the numbers on the other slides come from our own computations.

(Bucharest University of Economic Studies; Romanian Academy, Institute for Economic Forecasting)

Explore it visually

- **Transformer Explainer**: a GPT-2 model running in the browser: type a sentence, open one attention head to see the scores $\mathbf{q}^\top \mathbf{k}$, the softmax and the weighted sum of the values
  - Move the temperature to sharpen or flatten the next-token distribution.
- **LLM Visualization (B. Bycroft)**: a 3D walk through a small GPT, block by block: every matrix product of normalisation, attention and the feed-forward layer.
- **The Illustrated Transformer (J. Alammar)**: queries, keys and values drawn step by step for one sentence.
- **Attention variants (S. Raschka)**: from multi-head to grouped-query and latent attention, sliding windows and sparse attention: how recent models reduce the memory of keys and values.

Further reading and resources

- **Vaswani et al. (2017)**, "Attention is all you need": the original transformer.
- **Bishop & Bishop (2024)**, chapter 12: transformers, from attention to large language models.
- **The Illustrated Transformer** (J. Alammar): jalammar.github.io.
- **Hugging Face course**, fine-tuning encoders: huggingface.co/learn.
- **This course**: code, data, notebooks — github.com/danpele/NEURAL_BIZ.
- **Experiment yourself**: the notebook reproduces every chart and number of this lecture.

Quiz 1: scaling

- Why are attention scores divided by $\sqrt{d_k}$?

Answer

- With unit-variance entries the dot product $\mathbf{q}^\top \mathbf{k}$ has variance d_k
  - Dividing by $\sqrt{d_k}$ brings it back to 1, so the softmax does not saturate and its gradient does not vanish.

Quiz 2: positions

- What happens to a transformer without positional encoding?

Answer

- Self-attention is permutation-equivariant, so the model sees a bag of tokens
  - On our short tweets this cost only 0.9 points of macro-F1, but order matters for negation and long texts.

Quiz 3: pretraining

- Why does a transformer trained from scratch on 8 588 tweets lose to TF-IDF?

Answer

- It must learn language and task from few examples with 2 020 547 parameters
  - Pretrained FinBERT brings that knowledge and reaches 84.8 % after fine-tuning.

Quiz 4: encoder or decoder

- A bank wants to label 10,000 complaints a day by topic.
  - Encoder or decoder transformer?

Answer

- An encoder (BERT-type), fine-tuned on labelled complaints: it reads the whole text in both directions and outputs class probabilities, cheaply and consistently.

Quiz 5: attention maps

- Can an attention map justify a decision to a regulator?

Answer

- No: different maps can give the same prediction, and much attention goes to [SEP] (up to 59.3 % here)
  - Use attribution methods and tests instead.

References (1/2)

- Araci, D. (2019). FinBERT: Financial sentiment analysis with pre-trained language models. arXiv:1908.10063
- Ba, J. L., Kiros, J. R. & Hinton, G. E. (2016). Layer normalization. arXiv:1607.06450
- Bahdanau, D., Cho, K. & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*. arXiv:1409.0473
- Bishop, C. M. & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*, ch. 12. Springer. doi:10.1007/978-3-031-45468-4
- Clark, K., Khandelwal, U., Levy, O. & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. *BlackboxNLP Workshop*, 276–286. doi:10.18653/v1/W19-4828
- Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL*, 4171–4186. doi:10.18653/v1/N19-1423
- Efron, B. & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall. doi:10.1201/9780429246593
- Jain, S. & Wallace, B. C. (2019). Attention is not explanation. *Proceedings of NAACL*, 3543–3556. doi:10.18653/v1/N19-1357

References (2/2)

- Jheng, S.-L., Tak, R., Găman, Ș., Mazurencu-Marinescu-Pele, M. & Pele, D. T. (2026). Can foundation models manage risk? Zero-shot VaR and ES forecasting with conformal calibration in CEE markets. *Economic Computation and Economic Cybernetics Studies and Research*, 60(2), 267–290. doi:10.24818/18423264/60.2.26.14
- Loshchilov, I. & Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations*. arXiv:1711.05101
- Malo, P., Sinha, A., Korhonen, P., Wallenius, J. & Takala, P. (2014). Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65(4), 782–796. doi:10.1002/asi.23062
- Pele, D. T., Bolovăneanu, V., Lin, M.-B., Ren, R., Ginavar, A. T., Spilak, B., Andrei, A.-V., Toma, F.-M., Lessmann, S. & Härdle, W. K. (2026). In the beginning was the word: LLM-VaR and LLM-ES. SSRN Working Paper 5104383. doi:10.2139/ssrn.5104383
- Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI technical report. openai.com/research/language-unsupervised
- Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3), 1139–1168. doi:10.1111/j.1540-6261.2007.01232.x
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30. arXiv:1706.03762
- Wu, Y., Schuster, M., Chen, Z. et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144

Appendix A: gradient of the softmax and saturation

$$a_j = \frac{e^{s_j}}{\sum_l e^{s_l}}, \qquad \frac{\partial a_j}{\partial s_i} = a_j(\delta_{ij} - a_i)$$

$$J = \mathrm{diag}(\mathbf{a}) - \mathbf{a}\mathbf{a}^\top, \qquad \|J\| \to 0 \text{ when } \mathbf{a} \to \mathbf{e}_k$$

- If one score dominates, $\mathbf{a}$ is almost one-hot and every entry of J is close to zero: the queries and keys stop learning.
- Scores with standard deviation σ make domination likely when σ is large
  - Scaling by $\sqrt{d_k}$ fixes $\sigma = 1$ at initialisation.

(Vaswani et al., 2017)

Appendix B: parameters of a transformer block

$$\underbrace{4d^2 + 4d}_{\text{attention}} + \underbrace{8d^2 + 5d}_{\text{FFN}} + \underbrace{4d}_{\text{LN}} \approx 12d^2$$

- Attention: $W_Q, W_K, W_V, W_O \in \mathbb{R}^{d \times d}$ with biases
  - FFN: $W_1 \in \mathbb{R}^{4d \times d}$, $W_2 \in \mathbb{R}^{d \times 4d}$
  - Two layer norms with γ, β.
- BERT-base, $d = 768$: about 7 million per block, 85 million in 12 blocks, plus 23 million in the embedding table (as published).

(Devlin et al., 2019)

Appendix C: settings of the experiments

Model	Settings
TF-IDF	1–2-grams, minimum 2 documents, sublinear term frequency; logistic regression with balanced class weights, $C = 3$ chosen on validation
Small transformer	WordPiece tokens of FinBERT, 64 tokens; $d = 64$, 4 heads, 2 pre-norm blocks, FFN 128, dropout 0.1; AdamW $5 \cdot 10^{-4}$, batch 64, at most 20 epochs, patience 4
FinBERT fine-tuned	ProsusAI checkpoint, its own 3-class head re-mapped to bearish/bullish/neutral; AdamW $2 \cdot 10^{-5}$, linear decay, batch 32, 3 epochs; seeds 1–3
Frozen FinBERT	mean of the last hidden layer over the real tokens; logistic regression, $C = 0.03$
Hardware	Apple-silicon laptop, graphics processor, Metal Performance Shaders (MPS); total run 8.1 minutes