PRAGMA

Revolut’s Foundation Model

Amsterdam Banking Forum 2026

Can you guess the next word?

Yesterday, I went to the bank and the ATM swallowed my [...]

I immediately ordered a new one through my [...]

Can you guess the next (trans)action?

Created: 20-11-02 12:19:54
Type: card_payment
Direction: Out
Amount: 14.99
Currency: GBP
Fee: 0.00
Description: metal plan
MCC 6012
...

Created: 24-04-07 19:20:18
Type: communication
Channel: email
Product: stocks_shares_isa
Interact: Interacted

Created: 25-07-22 13:49:29
Type: app_event
Screen: junior_transfer

Why transactions are like language

PROPERTY	NATURAL LANGUAGE	FINANCIAL TRANSACTIONS
Sequential	Words in sentences	Transactions over time
Contextual	Meaning from surrounding words	Behaviour from surrounding transactions
Patterns	Grammar, idioms, style	Spending rhythms, habits
Anomalies	Grammatical errors	Fraudulent/unusual activity
Long-range deps	Document-level coherence	Seasonal patterns, life events

If masked language modelling can learn universal language representations (BERT),
masked transaction modelling can learn universal *behavioural* representations.

The traditional machine learning setup in banking

Each new model = **weeks/months** of feature engineering from scratch

For each ML model in production, teams must:

Understand the domain
Months of expertise gathering

Engineer features manually
Hundreds of hand-crafted aggregations

Validate features
Statistical testing, drift monitoring

Maintain pipelines
ETL jobs, feature stores, versioning

Typical hand-crafted features:

`count_transactions_last_7d` `ratio_international_txn`

`std_amount_by_hour_of_day` `avg_amount_merchant_category_30d`

`time_since_last_txn_seconds` `unique_merchants_90d`

The opportunity: Foundational Models

Slow iteration

Duplicated effort

Domain bottleneck

Can we learn *universal* representations of customer behaviour that transfer across all downstream tasks?

Inconsistency

Limited transfer

Scalability

The Shift: From features to representations

Before: Task-specific feature engineering

Raw data → Manual FE → Features_fraud

Raw data → Manual FE → Features_credit

Raw data → Manual FE → Features_aml

...repeat for every new model

After: Foundation model approach

Raw data → Tokenise → Pre-train FM → Universal embeddings

Universal embeddings → Fraud | Credit | AML | Cross-sell | Recurrents

- Train a single foundation model on ALL transaction data.
- Every downstream task benefits from shared learned representations.
- No manual feature engineering required.

Data model

Event streams

- Transactions
- Events from the app
- Trading events
- Communication events

Dataset

- 26M+ users
- 111+ countries
- 24B+ events
- 207B+ tokens
- 25 months of history

Tokenization

We represent each data point by three components: a semantic type (key), a value, and a temporal coordinate.

Created: 20-11-02 12:19:54

Type: card_payment

Direction: Out

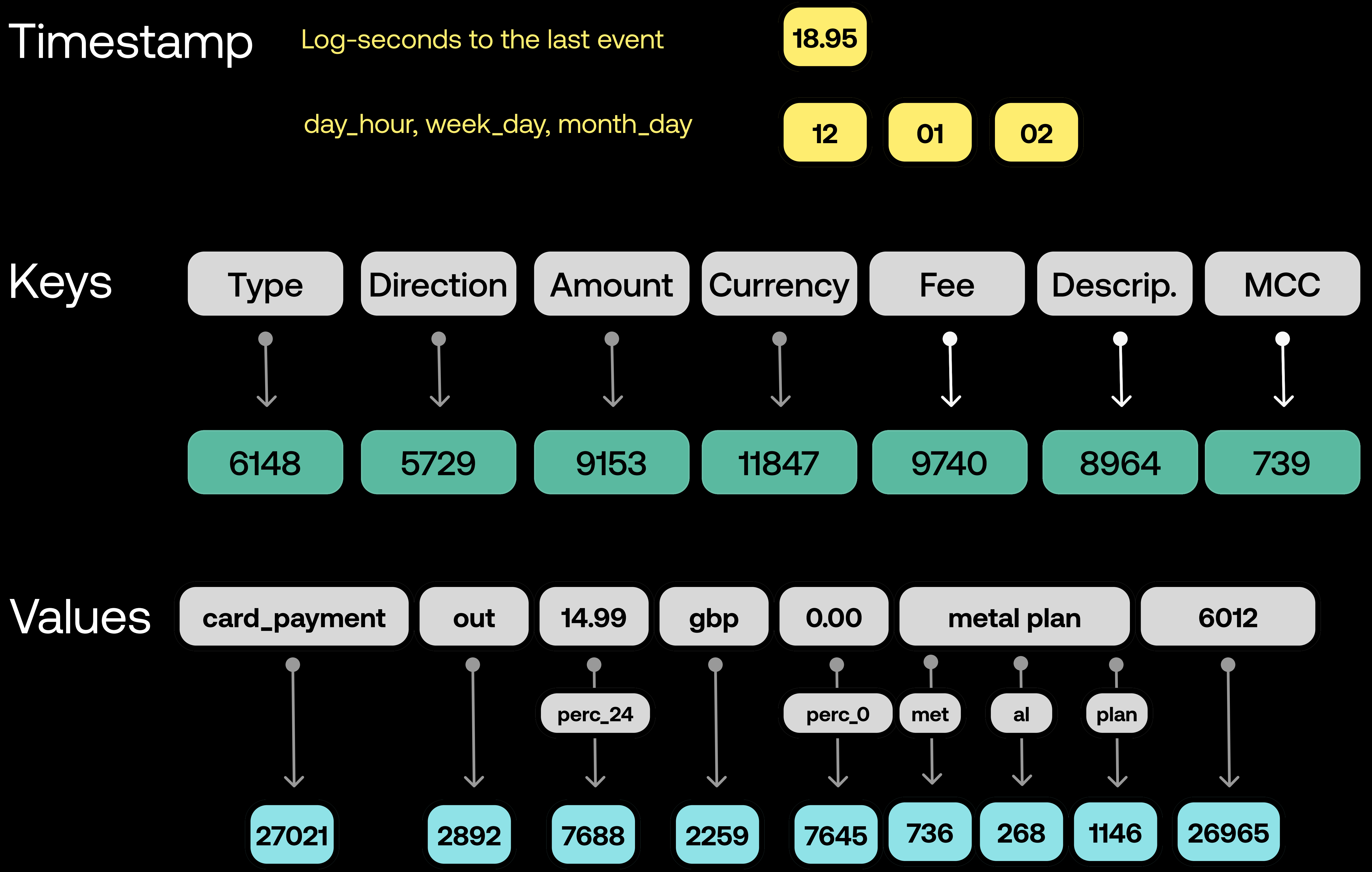
Amount: 14.99

Currency: GBP

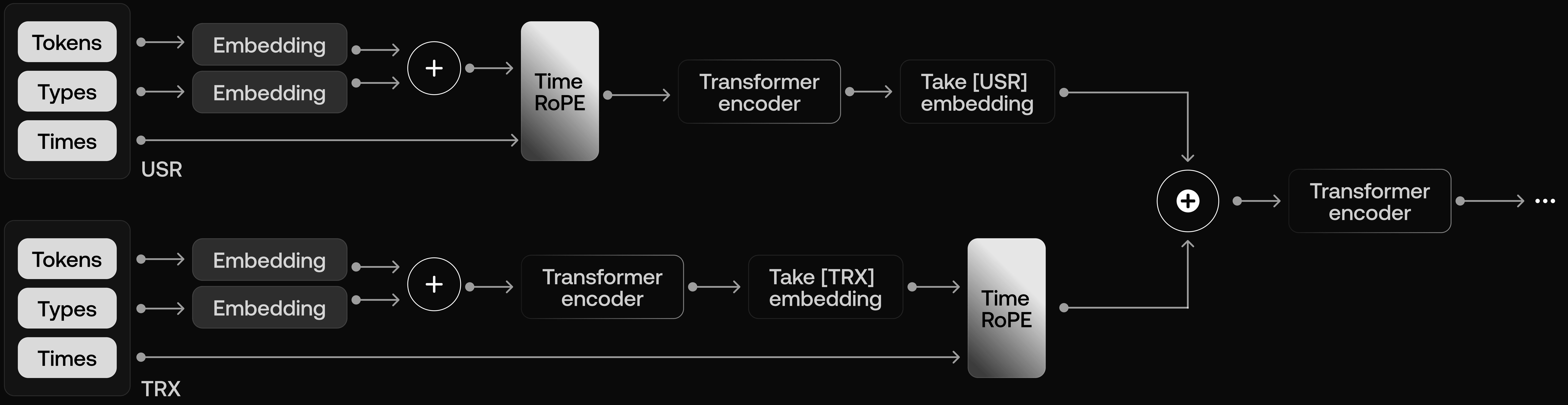
Fee: 0.00

Description: metal plan

MCC: 6012



Modelling



Legend

- Input tensors from USR and TRX
- Static RoPE with custom time-based positions
- Trainable embedding lookups
- Trainable transformer encoder layers
- Element-wise addition
- Concatenation

Pre-training: Masked Modeling

1. Randomly mask 15% of tokens in each transaction sequence
2. Replace with [MASK] token
3. Model predicts the original token from context

Example

Original	[card_payment]	[amount_q045]	[mcc_grocery]	[hour_14]	[dow_fri]
Masked	[card_payment]	[MASK]	[mcc_grocery]	[MASK]	[dow_fri]
Predict		^amount_q045		^hour_14	

Why it works

- Forces model to learn relationships between fields - “If a Friday pm grocery txn → amount likely in range X”
- Learns co-occurrence patterns - without any labels

LoRA: Efficient fine-tuning

The idea: Freeze all pre-trained weights. Add small low-rank matrices: $W_0x + \Delta Wx = W_0x + BAx$

Only train A and B (~1-2% of total parameters).

ASPECT	FULL FINE-TUNE	LORA
Trainable params	100%	~1-4%
Training cost	High	Low
Storage per task	Full model copy	Small adapter
Forgetting risk	High	Low
Switch tasks	Train new base model	Swap adapter

Five downstream applications

TASK	DESCRIPTION	BUSINESS IMPACT	RESULTS
Fraud detection	Real-time identification of fraudulent txns	Customer protection	+64.7% Recall
Credit risk	Assess credit worthiness from behavior	Lending decisions	+130.2 % PR-AUC
Cross-sell	Recommend relevant financial products	Revenue growth	+40.5% mAP
Recurrent transactions	Predict recurring payment patterns	Customer insights	+5.8% F1 (macro)
Credits engangement communications	Select the best way to communicate with the user	Revenue growth	+163.73% AUUC
Anti-money laundering	Detect suspicious transaction patterns	Regulatory compliance	- 47.1 % F(0.5)

All tasks benefit from the same pre-trained representations — patterns learned for fraud detection inform credit risk and vice versa.

Operational impact

Engineering efficiency

weeks → 0
Feature engineering time

months → days
New model prototype

maintenance ↓
Feature store maintenance reduced

Model development velocity

3-5x
Faster

↓60-80%
Reduction in production time

minimal build
For any ML engineer - no deep domain knowledge required

Infrastructure

1 model
One pre-trained model serves all downstream tasks

1 embed
Compute embeddings once, reuse across all tasks