

TRecos

TRecos: transformer-based next best action for financial services

Customer behavior is a sequence a snapshot can't capture

Executive Summary

Most next best action (NBA) engines in financial services read a snapshot of a customer's balance, tenure, and product holdings at a fixed point in time, unable to weight recency or sequence. Machine learning deterministic models are commonly used. We built TRecos on the latest transformer architecture, so it reads the whole sequence instead: the ordered history of what a customer did and when they did it. That ordered history is what lets it recommend the right product at the moment a customer is ready for it. TRecos has driven a 40% engagement lift vs. existing advanced recommendation engines and a 140% lift compared to early-stage engines. We deploy inside your environment, on your infrastructure, and it's production-ready in a matter of weeks.

TABLE OF CONTENTS

Executive Summary	01
What is the cost of a static snapshot?	03
Why do transformers fit Next Best Action?	04
Inside TRecos: a sequential model matched to each deployment	05
TRecos runs a live A/B test against your baseline, checked daily	06
A large U.S. payments firm replaced its rules-based engine with TRecos	07
Deployment in weeks, not months	07
The next-best-action use cases TRecos runs today	08
FAQs	09
The 30-day POC	10

What is the cost of a static snapshot?

Existing tabular machine learning (ML) and rules-based models are only able to see a fixed set of features, so they miss any sequencing and time-based context. Say a customer checks mortgage rates three times in a week. Their current NBA has no idea as it sees the same feature set it saw last month: income bracket, credit tier, current products, tenure. Rate-checking behavior either isn't captured at all, or it's flattened into a single binary flag that looks identical whether the last check was yesterday or six months ago.

That's the structural limit of tabular ML as applied to a recommendation engine. The model reads a row. A row has no concept of order or time. The model can be fed a feature called "days since last rate check," but that's just another static number to the model because it can't reason about a sequence of actions building toward a decision. And it can't weight recency, or separate signal from noise.



The result

The result is a recommendation engine that treats an urgent buying signal exactly the same as an idle one. An existing model might be able to surface a mortgage offer that would have fit the customer a few months ago, but it has no way of knowing their current situation has become more pressing. And if the offer comes in a week late, the customer may have already applied with a competitor. At the scale of a national bank or payments platform, that gap compounds across millions of customer interactions every day.

6 weeks

To first recommendation

CCPA

Compliant with full audit logging

Why do transformers fit Next Best Action?

Transformer-based models like TRecos use the same mechanism that lets a language model predict the next word based on everything that came before it to predict the next best action based on everything a customer has done before the present moment: every login, product page view, payment, and support call, parsed in the order it happened.

That ordering is fundamental to why transformer-based models are the right fit for an NBA engine. A transformer doesn't see "three mortgage rate checks" as a static event count. It sees three rate checks, spaced across nine days, following a jump in average balance, preceded by a change in login frequency. That sequence becomes the signal. Recency and momentum, the two things a tabular snapshot cannot structurally represent, are exactly what a sequential model is built to weight.

TRecos is a different class of model. Instead of a rules-based engine that encodes what a human analyst already believes matters, or a tabular ML model that finds patterns across static features, TRecos is a transformer-based engine that reads behavior the way it actually happens: over time, in a particular order, and with recent events weighted more heavily than distant ones.



**TRecos is
different.
It sees
sequence
where
other
models
see static
events.**

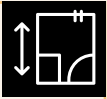
Inside TRecos: a sequential model matched to each deployment

TRecos' architecture starts with the input layer, where it converts every customer event (a login, a payment, a product view, a support call, a rate check, and many more) into an embedding. This numerical representation captures what the event was and where it sits in the sequence.

So a rate check three days ago and a rate check six months ago aren't just tagged with different time stamps. They occupy different positions in the embedding space entirely, which is what lets the model treat them as fundamentally different signals rather than the same feature with a different date attached.

Those embeddings feed into an attention mechanism, a core piece the model borrows from the structure of transformer-based large language models (LLMs). Attention lets the model weigh every event in a customer's history against every other event and decide which ones matter most to the prediction it must make.

A customer's mortgage rate checks might draw heavy attention when the model is scoring a home equity offer, and almost none when it's scoring a checking account upgrade. The model is learning, per customer and per decision, which parts of the sequence actually predict the outcome.



From there, the architecture splits by task. TRecos uses a shared sequence encoder, the part that reads and represents the customer's history, feeding into separate task heads:

- One task head scores cross-sell propensity
- Another one scores upsell readiness within a product category
- Yet another one scores churn risk.

We train each task head on the outcome it's responsible for, so the model scores a customer's representation differently depending on the question, without retraining the entire model for every new use case.

We select the best-fit configuration per deployment based on three things:

- 1 The client's data:** How much history exists, how clean the event logs are, and how many products are in scope.
- 2 The complexity of the use case:** A single-category upsell decision is a simpler prediction than a full-catalog NBA ranking across dozens of products.
- 3 The latency each channel requires:** A real-time recommendation surfaced inside a mobile banking session has a tighter latency budget than a batch-scored list generated overnight for an outbound campaign.

The model configuration reflects that last one with a lighter, faster-inference setup for the former and a deeper sequence model for the latter where there's more time to compute.

What stays constant across every configuration is the sequential core. The model reads ordered customer events through embeddings and attention rather than a flattened snapshot of static features, and every task head sits on top of a representation that already understands recency and momentum before it ever tries to predict an outcome.

TRecos runs a live A/B test against your baseline, checked daily

TRecos recommendations go to one segment and the incumbent system's go to another, and we compare outcomes on the metrics that already matter to you: conversion, engagement, and revenue per customer.

We check results daily using chi-square testing, which measures whether the observed difference between TRecos and the baseline is statistically real or just noise in a given day's traffic. This is a standing validation process that continues for the life of the deployment, so your risk and model governance teams have a continuous, testable answer to "is this actually working," not a single backtest from six months ago.

A large U.S. payments firm replaced its rules-based engine with TRecos

Swapping a snapshot-driven system for a sequential model reading full customer event histories, and validated through the same A/B, chi-square testing process, TRecos delivered a 40% engagement lift vs. existing advanced recommendation engines and a 140% lift when compared to other early-stage engines.

The gain came from the same mechanism described in the sections above: TRecos weighted recent behavior and reasoned about sequence in a way the rules-based system had no way to replicate. Nothing about the underlying customer base or product catalog changed. The model reading the data changed.



40%

engagement lift vs. advanced recommendation engines

140%

engagement life vs. early-stage recommendation engines

Deployment in weeks, not months

TRecos deploys inside your environment, so customer data never leaves it, and the model runs on your existing infrastructure. It connects through the data warehouse, CDP, or core banking system already in place, typically read-only by default.

This means there is no need for rip-and-replace. TRecos sits upstream of the campaign tools, offer engines, and channels you already run, feeding them ranked next-best-action recommendations rather than requiring a new platform to sit on top of. From raw data to production recommendations, deployment takes weeks, not months, and the resulting recommendations are exportable to whatever channel you're already using to reach your customers.

YOUR ENVIRONMENT

Customer data never leaves



We build model governance into the deployment, so every recommendation ships with a feature-level rationale and an audit trail. Versioning, approvals, champion/challenger testing, and drift monitoring come standard in every configuration, which matters as much to a risk officer reviewing the model as it does to the team measuring its lift.

The next-best-action use cases TRecos runs today

We deploy TRecos across several categories of recommendation inside financial institutions:

Credit card cross-sell. Identifying which existing customers are moving toward a credit product, and which specific card fits the behavioral pattern they're showing.

Loans and consumer credit upsell. Surfacing customers ready to move up within a credit category, mortgage, personal loan, or line of credit, based on sequence rather than a static eligibility check.

Retention and churn signals. Detecting the behavioral pattern of a customer drifting toward attrition early enough for a retention action to still be effective.

Deposits and everyday banking products. Driving uptake of debit cards, digital wallets, and checking accounts by reading the relationship-building signals tabular snapshots miss.

Wealth and merchant offer ranking. Ranking the single best next best action for a customer across an institution's full product range, rather than scoring each product line separately.

Frequently asked questions

?

What is a transformer-based next best action engine?

It's a recommendation system that reads a customer's full, sequentially ordered history of behavior (rather than relying on their current attributes alone) to predict what they need next. The transformer architecture weights recent and sequential events, the same mechanism used by LLMs to predict the next word in a sentence.

?

How is it different from a rules-based or traditional ML recommendation model?

Rules-based and tabular ML models read a static snapshot of features at one point in time. They are unable to represent recency or momentum. TRecos reads the ordered sequence of events, which is why it catches buying signals that snapshot models miss.

?

Does TRecos require moving our data to a vendor platform?

No. TRecos deploys inside your tenant, connecting through the existing data warehouse, CDP, or core banking system. Customer data stays inside your environment throughout, typically accessed as read-only.

?

How long does it take to get TRecos to production?

Deployment takes a matter of weeks, not months, and runs on your existing infrastructure. You can validate the approach on your own data through a **30-day** proof of concept before committing to a full rollout.

?

What results has TRecos delivered?

At a large U.S. payments firm, TRecos delivered a **40%** engagement lift vs. existing advanced recommendation engines and a **140%** lift when compared to other early-stage engines. We validated this deployment through ongoing A/B testing against the prior baseline.

The 30-day POC

The lowest risk way to see whether TRecos outperforms an existing recommendation stack is to run it against real customer data. We scope a 30-day proof of concept inside your environment, using your data and your customer base so the results reflect your institution's actual behavior patterns rather than a generic benchmark.

There's no data migration and no commitment beyond the 30 days. It's a direct comparison: your existing system's recommendations vs. TRecos on the same customers, measured against the same metrics you're already tracking.

A recommendation is only as good as what the model is built to see. Give it a snapshot of a customer's life and the model is accurate on average but wrong in the high-intent moments where timing decides the outcome. Give it the whole sequence of customer events and you'll find the timing problem that has been costing your institution money for years starts to close up.

TRecos exists because that second option is now buildable within your walls, on your data, on your timeline, and without needing to rebuild your stack. The technology has caught up to the problem, all that's left is to decide how soon you want it in production for your customers.

Key Takeaways



Tabular and rules-based NBA engines read a static snapshot and are not structurally able to weight recency or sequence like a transformer-based model can, no matter how often they're retrained.

TRecos converts customer events into embeddings, applies attention to weigh which events matter most per decision, and scores separate outcomes through dedicated task heads sitting on a shared sequence encoder.

TRecos uses a multi-technique architecture, selecting the best-fit model per deployment based on data, use case, and latency requirements.

At a large U.S. payments firm, TRecos delivered a 40% engagement lift vs. existing advanced recommendation engines and a 140% lift when compared to other early-stage engines.

TRecos deploys inside your environment, on your infrastructure, in weeks with no rip-and-replace and no data ever leaving your systems.

Want to see it run on your data? Email us to book a discovery session where we will walk through where TRecos fits in your NBA workflow.

contactus@sirusai.com

sirusai.com/trecos