Behavioral Intelligence Platforms:

From Event Streams to Autonomous Insight via Probabilistic Journey Graphs, Behavioral Knowledge Extraction, and Grounded Language Generation

Arun Patra*
Journium, Inc.
arun@journium.app

Bhushan Vadgave
Journium, Inc.
bhushan@journium.app

March 2026

Abstract

Contemporary product analytics systems require users to pose explicit queries—writing SQL, configuring dashboards, or constructing funnels—before any insight can surface. This pull-based paradigm imposes a dual bottleneck: it demands both domain knowledge and technical fluency from practitioners who must know, in advance, which questions to ask. We argue that the next generation of behavioral analytics must invert this model, shifting from passive data stores that answer queries to active intelligence systems that continuously monitor, detect, and narrate behavioral phenomena without prompting.

We present the Behavioral Intelligence Platform (BIP), a system architecture and formal framework that transforms raw event streams into automatically generated, evidence-backed insights. BIP introduces four tightly integrated layers: (1) a Normalization and State Derivation (NSD) stage that standardizes raw events and maps them to a multi-level semantic state hierarchy; (2) a Behavioral Graph Engine (BGE) that models user journeys as absorbing Markov chains, computing transition probabilities, removal effects, and path quality metrics; (3) a Behavioral Knowledge Graph (BKG) combined with a Detector System (DS) that reifies graph outputs into a queryable triple-store of grounded behavioral facts and autonomously identifies behavioral phenomena; and (4) a Grounded Language Layer (GLL) that constrains large language model (LLM) output to verified facts from the BKG, producing faithful narrative insights at scale.

We formalize the Behavioral Intelligence Problem, define a taxonomy of detectors for autonomous insight generation (activation drivers, drop-off clusters, behavioral regressions, segment divergence, unexpected loops), and propose a composite interestingness score for prioritizing insights under bounded attention. We discuss the key design principles—grounded-first computation, semantic state abstraction, time-aware snapshots, and segment-aware analysis—and characterize the fundamental trade-offs between completeness, causal interpretability, and computational cost. We define evaluation criteria for correctness, latency, and insight quality, and characterize the fundamental challenges of assessing behavioral intelligence systems

*Code, simulation scripts, and datasets: https://github.com/journium/journium-research.

— challenges that are unique to unsupervised, push-based insight generation and that motivate an open evaluation agenda.

Keywords: behavioral analytics, user journey modeling, Markov chains, knowledge graphs, automatic insight generation, proactive analytics, large language models, product analytics, retrieval-augmented generation, event stream processing

Contents

1 Introduction

The canonical workflow of a product analyst begins with a question. The analyst opens a dashboard, constructs a query, examines a chart, and forms a hypothesis—then iterates. This interactive, pull-based model has historically underpinned virtually every analytics platform in commercial use, from Mixpanel and Amplitude to Google Analytics and Heap. The user is assumed to be an active driver of inquiry; the system is a passive responder.

This assumption is increasingly untenable. Product teams face an ever-expanding corpus of instrumented behavior—hundreds of event types, thousands of user properties, dozens of defined funnels—against which the space of meaningful questions is combinatorially vast. Under the pull model, most of this behavioral signal goes unexamined. Insights that are not explicitly sought are not found. Critical regressions in activation flows go undetected until they manifest in lagging revenue metrics. Segment-specific friction points accumulate invisibly behind aggregate conversion numbers. Fast-moving product cycles demand a pace of inference that manual exploration cannot sustain.

We observe a structural mismatch: the volume and velocity of behavioral data have grown substantially, while the human capacity for directed inquiry has not. The solution is not more powerful query engines or more expressive visualization tools—it is a paradigm shift from pull-based to push-based intelligence, where the analytics system takes epistemic initiative.

This paper formalizes and addresses the *Behavioral Intelligence Problem*: given a continuous stream of product events across a population of users or accounts, automatically detect, rank, and narrate behavioral phenomena of potential significance—without requiring the practitioner to specify what to look for.

This problem is distinct from, and more ambitious than, existing adjacent work. Automated data exploration (ADE) systems [7, 9] automate the exploration of tabular datasets but lack the journey-centric, temporal, and segment-aware structure of behavioral analytics. Anomaly detection systems identify statistical deviations but do not model user journeys, cannot reason about path structure, and produce point-in-time alerts rather than behaviorally coherent narratives. Even recent benchmarks for natural language query interfaces to structured data [5] presuppose that the user poses a question. We seek a system that generates questions and answers them simultaneously.

Our contributions are as follows:

- **Problem formalization.** We formally define the Behavioral Intelligence Problem, characterizing the input (event streams), the output (a ranked feed of evidence-backed insight objects), and the objectives (high interestingness, statistical validity, actionability, and faithfulness).
- **Multi-level state model.** We introduce a three-level state hierarchy (raw event, semantic, lifecycle) and formalize state derivation as a rule-based mapping that enables meaningful journey abstraction while preserving traceability to raw events.
- **Absorbing Markov chain journey model.** We model user journeys as absorbing Markov chains over a derived state space and derive closed-form expressions for conversion probability, expected journey length, and state removal effects.
- **Behavioral Knowledge Graph.** We define a typed fact schema—behavioral triples of

the form (subject, predicate, object) with associated evidence payloads and confidence scores—that serves as an auditable intermediate representation between numerical computation and language generation.

- **Detector taxonomy.** We taxonomize the class of behavioral phenomena detectable by deterministic detectors and propose an interestingness scoring framework for prioritizing the resulting insight feed.
- **Grounded Language Layer.** We propose an architecture for constraining LLM-generated narratives to verified knowledge graph facts, separating numerical computation from linguistic expression and systematically preventing hallucination in analytics narratives.
- **Design principles and trade-off analysis.** We characterize four core design principles for behavioral intelligence systems and analyze the trade-offs they entail, with reference to existing literature.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 formally defines the Behavioral Intelligence Problem. Section 4 describes the BIP architecture. Section 5 develops the formal foundations. Section 6 describes the detector system. Section 7 describes the Grounded Language Layer. Section 8 discusses evaluation. Section 9 analyzes limitations. Section 10 concludes.

2 Related Work

2.1 Automated Data Exploration and Insight Generation

The problem of automatically generating insights from tabular data has received sustained attention in the database and visualization communities. Law et al. [7] characterize the space of automated data insights, identifying properties such as novelty, deviation, and coverage as criteria for interestingness. DataShot [17] generates fact sheets from tabular data by enumerating a predefined set of statistical patterns (extrema, trends, comparisons) and ranking them by interestingness. MetaInsight [9] discovers structured knowledge from multidimensional data by identifying common patterns across subspaces via an optimal partition scheme.

More recent work has explored LLM-mediated insight generation. The QUIS system [11] uses a two-module architecture in which a question-generation module produces statistically-informed questions about a dataset and a downstream module answers them, iterating until insight quality converges. InsightBench [13] provides a benchmark for evaluating end-to-end data analysis agents. InsightPilot [10] proposes an LLM-empowered automated data exploration system. These systems target general tabular data and do not address the journey-centric, temporal, and segment-structured nature of product behavioral data.

A key distinction between prior ADE work and the behavioral intelligence setting is the role of time and sequence. Tabular ADE systems treat records as independent observations; behavioral data is fundamentally sequential and causal, with the ordering and structure of events encoding user intent, friction, and outcome probability in ways that cross-sectional summary statistics cannot capture.

2.2 Markov Models for User Journey Analysis

Markov chain models have a long history in modeling sequential user behavior, including customer journey attribution across marketing touchpoints. Harbich et al. [4] propose a mixture of Markov models for customer journey map discovery. The MDPI Customer Behaviour Hidden Markov Model [16] extends this to e-commerce with hidden state representations.

The absorbing Markov chain formalism provides a natural foundation for computing expected journey lengths and outcome probabilities in the presence of uncertain paths. In an absorbing Markov chain with transient states T and absorbing states A, the fundamental matrix $N = (I - Q)^{-1}$ gives the expected number of times the chain visits each transient state before absorption, enabling closed-form derivation of conversion probabilities and expected remaining steps [6].

Existing journey analysis tools—Mixpanel's Flows, Amplitude's User Paths, Heap's Path Analysis—offer UI-level path visualization but do not expose the underlying probabilistic model, do not compute removal effects, and require manual interpretation.

2.3 Process Mining

Process mining [15] is a discipline concerned with discovering process models from event logs in enterprise workflows. Customer journey analysis has been studied through a process mining lens [1], leveraging CJM-structured event logs to assess journey duration and quality.

BIP is conceptually related to process mining but addresses a different setting: consumer product events rather than enterprise process logs, behavioral heterogeneity rather than conformance to a predefined process model, and automated insight generation rather than interactive exploration.

2.4 Knowledge Graphs and Grounding

Knowledge graphs represent entities and their relationships as typed triples (subject, predicate, object), providing a structured representation suitable for both machine reasoning and natural language generation. The integration of LLMs with knowledge graphs—under the umbrella of Knowledge Graph Retrieval-Augmented Generation (GraphRAG) [8, 18]—has emerged as a dominant paradigm for grounding language model outputs in verified external knowledge.

Existing GraphRAG systems such as SubgraphRAG [8] retrieve relevant subgraphs and use them as context for LLM inference. These systems ground the LLM in pre-existing knowledge bases; BIP's Behavioral Knowledge Graph is dynamically constructed from live behavioral data and populated by deterministic detectors, making the grounding relationship tighter and more auditable.

Pan et al. [12] describe KGs as a 'cognitive middle layer' between raw input and LLM reasoning. BIP instantiates this principle in the specific domain of behavioral analytics.

2.5 Proactive and Augmented Analytics

The concept of augmented analytics was identified as a key trend by Gartner [2] and has since been extended toward proactive analytics paradigms [3]. Commercial systems such as ThoughtSpot Spotter, Salesforce Einstein Discovery, and Microsoft Power BI Copilot offer varying degrees of automated insight generation, primarily targeting anomaly detection and natural language query interfaces.

These systems typically operate on pre-aggregated metrics and lack the journey-level behavioral modeling that characterizes BIP.

3 Problem Formulation

3.1 Setting

Let $E = \{e_1, e_2, \ldots, e_n\}$ denote a temporally ordered stream of events, where each event $e_i = (\text{actor\_id}, \text{event\_name}, \text{timestamp}, \text{properties}, \text{context})$ describes a discrete action taken by an actor (user or account) at a specific moment. The stream is partitioned by organization and project, and events are associated with actors whose identity may be resolved across anonymous and authenticated sessions.

A segment S is a predicate over actor properties defining a cohort: $S(a) = 1$ iff actor a satisfies the segment criteria. Let $J = \{S_j \mid j = 1..K\}$ be a defined set of segments. We are additionally given a set of terminal states T representing outcome labels assigned to actors at the end of an observation window:

$$T = \{\text{converted},\ \text{churned},\ \text{inactive},\ \text{retained},\ \text{dropped\_off}\}$$

3.2 The Behavioral Intelligence Problem

Given a continuous event stream E, a set of semantic state definitions Φ, journey definitions Δ, segment definitions J, and outcome labels T, the Behavioral Intelligence Problem is to produce a ranked feed $I = \{(\text{insight\_type}, \text{content}, \text{confidence}, \text{evidence}, \text{timestamp})\}$ such that:

- **Validity:** Every claim in an insight object can be traced to a deterministic computation over E with stated statistical support.
- **Interestingness:** Insights are ranked by a composite score combining statistical significance, magnitude of effect, actionability, and novelty relative to prior observations.
- **Faithfulness:** Narrative descriptions of insights do not introduce claims not supported by the underlying evidence payload.
- **Coverage:** The system generates insights across the full space of detectable behavioral phenomena, not only those corresponding to user-specified queries.
- **Timeliness:** Insights reflect behavioral changes within a latency bound appropriate to their actionability.

3.3 Formal Objectives and Non-Goals

BIP does not claim to solve causal inference. The behavioral facts it produces reflect statistical associations that may or may not reflect causal relationships. The system uses association predicates (`increases_probability_of`, `associated_with`, `more_common_in`) rather than causal predicates (`causes`, `leads_to`) in its knowledge representation.

BIP also does not target real-time streaming at sub-second latency, nor fully learned sequence models such as transformer-based next-event predictors.

4 System Architecture

4.1 Overview

BIP is organized as a layered pipeline with four principal stages, each producing a structured artifact consumed by the next. Figure 1 shows the high-level data flow.

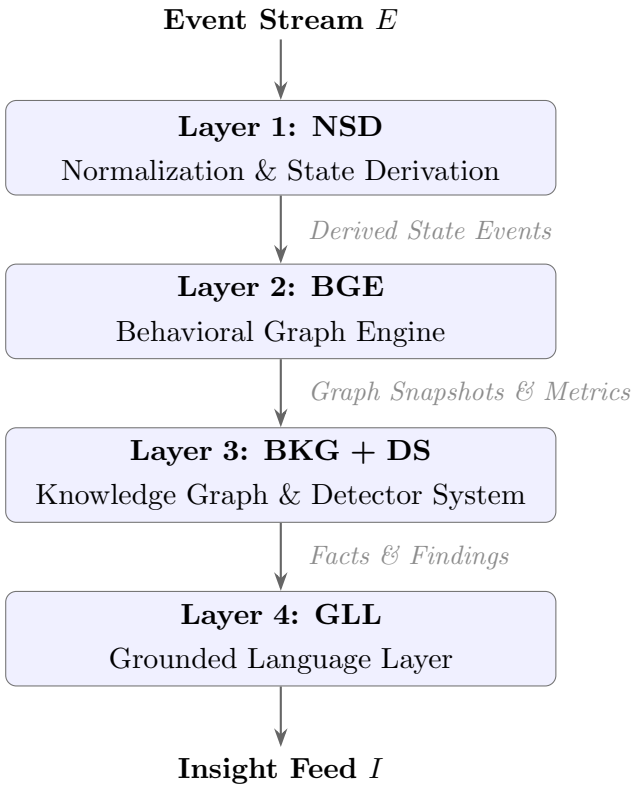

Figure 1: BIP four-layer data-flow architecture. Each layer produces a structured artifact consumed by the next: NSD derives a semantic state hierarchy from raw events; BGE builds absorbing Markov chain snapshots; BKG+DS reify graph metrics into grounded facts and emit typed findings; GLL generates faithful narrative insights constrained to verified facts.

4.2 Layer 1: Normalization and State Derivation

The normalization stage performs five operations on raw events: (i) deduplication by `event_id` with configurable idempotency window, (ii) bot and health-check filtering via

User-Agent and behavioral heuristics, (iii) identity resolution that stitches anonymous and authenticated sessions using a deterministic graph traversal over aliasing events, (iv) late arrival handling with a configurable ingestion lag tolerance, and (v) timestamp normalization to UTC with sub-millisecond precision.

Following normalization, raw events are mapped to derived state events via a three-level state hierarchy. At the `raw_event` level, state identity is the event name itself. At the semantic level, states are defined by rule-based predicates over event properties. At the lifecycle level, states represent coarse user lifecycle positions—`new_user`, `activated`, `paying`, `churned`, `retained`—derived from a combination of event history and time windows.

This three-level abstraction reduces state space explosion: a product with 200 raw event types typically yields 15–30 meaningful semantic states and 5–8 lifecycle states.

4.3 Layer 2: Behavioral Graph Engine

The BGE aggregates derived state events into a directed weighted graph $G = (V, E, W)$ where V is the set of states, $E \subseteq V \times V$ is the set of observed transitions, and $W : E \to \mathbb{R}^+$ assigns transition counts. The BGE materializes the following metrics for each graph snapshot: (i) per-edge transition probabilities, (ii) per-state reach rates, outcome conversion probabilities, and expected remaining steps under the absorbing Markov chain model, (iii) top-N materialized paths ranked by occurrence and conversion efficiency, and (iv) segment-conditional versions of all of the above.

4.4 Layer 3a: Behavioral Knowledge Graph

The BKG converts BGE outputs and detector findings into a typed triple-store of behavioral facts. Each fact takes the form:

$$\begin{aligned} \mathit{fact} = (\,&\text{subject},\ \text{predicate},\ \text{object},\ \text{confidence},\\ &\text{validity\_window},\ \text{evidence\_payload},\ \text{provenance}\,) \end{aligned}$$

Predicates are drawn from a closed vocabulary of behavioral relations:

- `transitions_to`, `increases_probability_of`, `is_activation_driver_for`
- `is_dropoff_point_for`, `diverges_from`, `regressed_after`, `changed_after`

The closed predicate vocabulary constrains the knowledge graph to a semantically well-defined ontology and ensures that retrieval patterns for the GLL remain tractable.

Each fact is bound to the snapshot from which it was derived, giving the system a complete lineage from raw events through derived states through graph metrics through facts.

4.5 Layer 3b: Detector System

Detectors are deterministic analysis modules that consume BGE snapshot outputs and emit typed findings and facts. The detector system runs after each snapshot build and is extensible: new detectors can be added without modifying the BGE or BKG schema.

4.6 Layer 4: Grounded Language Layer

The GLL translates structured findings and facts into natural language insights via a two-stage process: (i) a retrieval stage that constructs a fact bundle—the minimal set of BKG facts supporting a given finding—and (ii) a generation stage that prompts an LLM with the fact bundle and a system prompt enforcing association-not-causation language. The LLM is explicitly prohibited from inventing numerical values, computing novel statistics, or asserting causal relationships.

5 Formal Foundations

5.1 Absorbing Markov Chain Journey Model

We model a journey definition δ as an absorbing Markov chain $M_\delta = (\mathcal{S}, \mathcal{A}, Q, R)$ where $\mathcal{S}$ is the set of transient (non-terminal) states, $\mathcal{A}$ is the set of absorbing (terminal) states, Q is the $|\mathcal{S}| \times |\mathcal{S}|$ sub-stochastic matrix of transition probabilities among transient states, and R is the $|\mathcal{S}| \times |\mathcal{A}|$ matrix of transition probabilities from transient states to absorbing states.

Transition probabilities are estimated from observed journey instances. For a pair of states (s_i, s_j), the empirical transition probability is:

$$P(s_j \mid s_i) = \frac{\text{count}(s_i \to s_j)}{\sum_k \text{count}(s_i \to s_k)} \tag{1}$$

where the denominator sums over all states, including absorbing states. This satisfies the row-stochastic property: for each transient state s_i,

$$\sum_j P(s_j \mid s_i) + \sum_t P(t \mid s_i) = 1 \tag{2}$$

Transition probabilities are estimated over a fixed analysis window, implicitly assuming behavioral stationarity within that window. This assumption is acknowledged to be an approximation; Section 9 discusses non-stationarity as a direction for future work.

Figure 2 shows a concrete six-state funnel modeled as an absorbing Markov chain. Four transient states (`sign_up`, `feature_used`, `import_data`, `invite_teammate`) lead to two absorbing states (`converted` and `dropped_off`). The dashed back-edge from `feature_used` to `sign_up` illustrates re-visitation, which is fully representable in the model.

5.1.1 Fundamental Matrix

The fundamental matrix

$$N = (I - Q)^{-1} \tag{3}$$

provides the expected number of times the chain visits transient state s_j when starting from transient state s_i before absorption. The matrix N exists and is finite when the

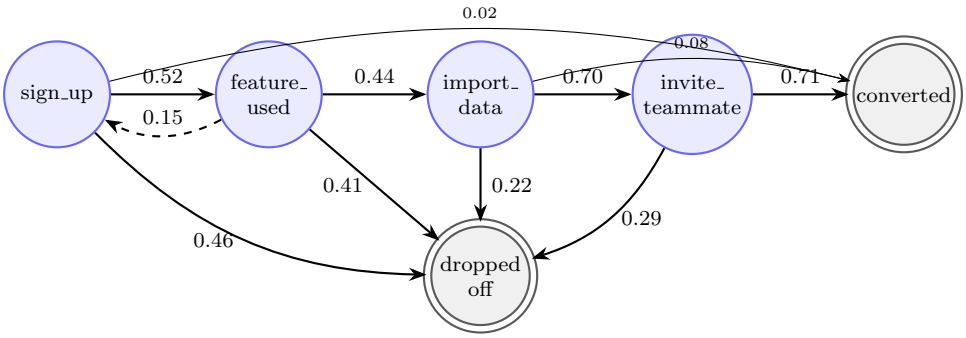

Figure 2: Example six-state funnel as an absorbing Markov chain. Single-bordered circles are transient states ($\mathcal{S}$); double-bordered circles are absorbing states ($\mathcal{A}$). Edge labels are empirical transition probabilities; each row sums to 1 across all outgoing edges (thin arrows indicate low-probability direct conversions). The dashed back-edge represents re-visitation from `feature_used` to `sign_up`.

chain is absorbing—a condition we enforce by design via the terminal state requirement in journey definitions.

5.1.2 Absorption Probabilities

The probability of being absorbed by terminal state t when starting from transient state s_i is given by the (i, t) entry of the absorption probability matrix:

$$B = N \cdot R \tag{4}$$

In the product analytics context, $B(s_i, \text{converted})$ is the probability of conversion when **starting from** transient state s_i — equivalently, the long-run fraction of journeys that reach `converted` among those that begin at s_i.

In the empirical product context we additionally track $P(\text{converted} \mid \text{reached}(s))$ — the fraction of observed journeys that converted among all that *visited* s at any point. Figure 3 plots this empirical conditional and its complement for each transient state. For sequential funnels with low re-visitation, $P(\text{converted} \mid \text{reached}(s))$ closely approximates $B(s, \text{converted})$; in the presence of significant looping, the two measures may diverge.

States with a large gap between $P(\text{converted} \mid \text{reached}(s))$ and $P(\text{converted} \mid \neg\,\text{reached}(s))$—`feature_used`, `import_data`, and `invite_teammate` in this example—are strong candidate activation drivers.

5.1.3 Expected Remaining Steps

The expected number of steps until absorption from transient state s_i is given by the row sum of N:

$$t_i = \sum_j N_{ij} \tag{5}$$

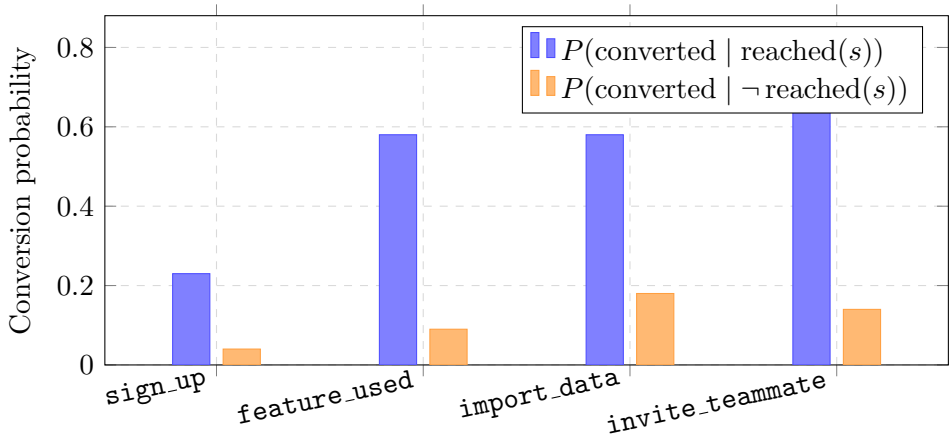

Figure 3: Absorption probabilities B (Eq. (4)) computed from synthetic simulation data. Blue bars show the probability of conversion given the user visited state s; orange bars show conversion probability among users who never visited s. Note that `import_data` achieves a lift of $0.58/0.18 \approx 3.2\times$, consistent with the fact bundle example in Section 7.

This quantity serves as a proxy for journey friction and is used in the drop-off detector and path quality scoring.

5.2 State Reach Rate and Conversion Lift

The reach rate of state s in a journey is the fraction of journey instances that include at least one visit to s:

$$\text{reach\_rate}(s) = \frac{|\{j : s \in \text{states}(j)\}|}{|\text{journeys}|} \tag{6}$$

The conversion lift of state s with respect to terminal outcome t measures the increase in conversion probability associated with visiting s:

$$\text{lift}(s, t) = \frac{P(t \mid \text{reached}(s))}{P(t \mid \neg\,\text{reached}(s))} \tag{7}$$

For states where $P(t \mid \neg\,\text{reached}(s)) = 0$, lift is undefined; the platform emits a special `necessary_for_conversion` finding. States with high reach rate and high lift are candidate activation drivers.

5.3 Removal Effect

The removal effect of state s with respect to terminal outcome t is the decrease in overall conversion rate when s is removed from the journey graph. Removal is modeled by: (i) deleting all edges into and out of s from the transition graph, (ii) re-normalizing transition probabilities from predecessor states of s, and (iii) recomputing the absorption probability matrix B' under the modified graph:

$$\text{removal\_effect}(s, t) = B(\text{start}, t) - B'(\text{start}, t) \tag{8}$$

A high removal effect indicates that the state lies on the dominant conversion paths in the graph. Note that removal effect is not equivalent to causal effect; it is a structural property of the observed graph.

Figure 4 ranks non-start transient states by removal effect. `import_data` has the highest structural impact ($\Delta \approx 0.14$): as the sole gateway to `invite_teammate`, removing it eliminates the dominant conversion path.

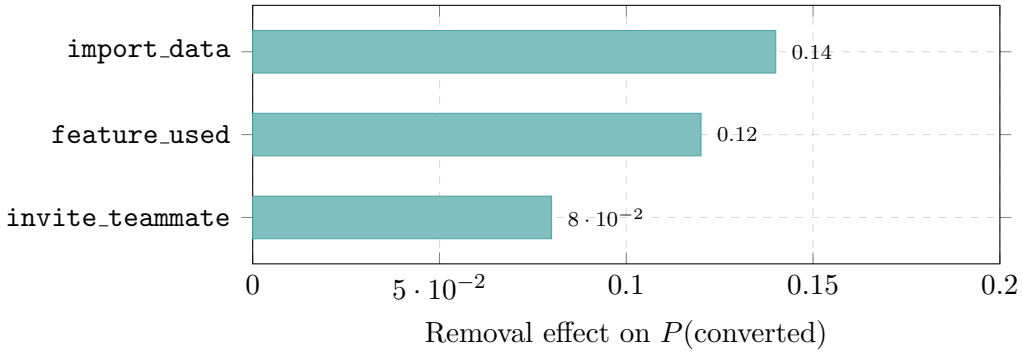

Figure 4: Removal effect ranking (Eq. (8)) for the example funnel. Each bar shows the decrease in overall P(converted) when the corresponding state is removed from the journey graph and transition probabilities are re-normalised. `import_data` is the structurally most critical state, as it is the sole gateway to `invite_teammate` — the highest-converting transient state.

The computational cost of removal effect is $O(|S|^3)$ per state per snapshot due to matrix inversion. For large state spaces, we apply it selectively to candidate states identified by a combined score of reach_rate $\times$ lift $\times$ sample_size $> \tau$.

5.4 Temporal Change Detection

For each edge (s_i, s_j) in the behavioral graph, we compute the delta in transition probability between consecutive snapshots:

$$\Delta P(s_i, s_j) = P_t(s_j \mid s_i) - P_{t-1}(s_j \mid s_i) \tag{9}$$

Statistical significance of changes is assessed using a two-proportion z-test under the null hypothesis that the two proportions are equal.

For release-linked change detection, the comparison window is anchored to the release marker. When multiple release markers fall within the comparison window, attribution is marked as ambiguous.

5.5 Interestingness Scoring and Insight Prioritization

We define a composite interestingness score adapted to the behavioral analytics setting:

$$\text{score}(f) = \alpha \cdot \text{significance} + \beta \cdot \text{magnitude} + \gamma \cdot \text{reach} + \omega \cdot \text{actionability} + \varepsilon \cdot \text{novelty} \tag{10}$$

where:

- **significance** is the statistical confidence of the finding ($1 - p$-value), penalized for small sample sizes.
- **magnitude** is the normalized effect size: $|\text{lift} - 1|$ for conversion-related findings, $|\Delta P|/P_{t-1}$ for regression findings, normalized to $[0, 1]$.
- **reach** is the fraction of the user population affected by the finding.
- **actionability** is a heuristic score that favors findings for which product or engineering actions are known to exist.
- **novelty** measures the recency of the finding relative to prior snapshots.

The weights $(\alpha, \beta, \gamma, \omega, \varepsilon)$ are configurable per deployed monitoring context. We acknowledge that the definition of interestingness is inherently subjective [14].

Table 1 illustrates the scoring algorithm applied to five synthetic findings from a representative product funnel. The prioritisation is well-behaved: the high-significance, high-reach activation driver (F002) ranks first; the statistically weak, small-sample drop-off cluster (F005) ranks last.

Table 1: Composite interestingness scores for five synthetic detector findings (weights: α=0.30, β=0.25, γ=0.20, ω=0.15, ε=0.10). Scores computed from synthetic simulation data by `interestingness_scoring.py`.

Rank	ID	Detector	Finding (summary)	Score
1	F002	ActivationDriver	`feature_used` is activation driver for `converted` (lift 4.2×)	0.845
2	F001	TemporalRegression	`email_verified`→`profile_complete` drop-off +18 pp post-v2.3	0.736
3	F004	SegmentDivergence	Mobile conversion 22 pp below desktop	0.602
4	F003	UnexpectedLoop	Loop: `profile_complete`→`sign_up` (8% of users)	0.252
5	F005	DropOffCluster	Drop-off at `sign_up`→abandoned (low significance, N=85)	0.054

5.6 Confidence Scoring

Each behavioral fact carries a confidence score in $[0, 1]$ computed as:

$$\text{confidence} = \sigma\left(a \cdot z_{\text{score}} + b \cdot \log \frac{n}{n_{\min}} + c \cdot |\text{effect}|\right) \tag{11}$$

where σ is the logistic function, z_{score} is the z-statistic, n is the sample size, $n_{\min}$ is the configured minimum threshold, and (a, b, c) are coefficients calibrated against historical annotation data (deployment-specific). Confidence is presented to end users as a qualitative label (High, Medium, Low).

6 Detector System

6.1 Design Principles

The detector system operationalizes the claim that behavioral intelligence is generated, not queried. Each detector implements a specific detection hypothesis—a pattern in the behavioral graph that, if present and statistically supported, corresponds to a finding of a defined type. Detectors are deterministic: given the same snapshot inputs, they always produce the same outputs.

Detectors are versioned. When detector logic changes, previously produced findings are not modified; future snapshots are processed with the new version.

6.2 Taxonomy of Detectors

6.2.1 Activation Driver Detector

The activation driver detector identifies states strongly associated with conversion to a target terminal state. It: (1) computes reach rate and lift for all states, (2) filters to candidate states with reach_rate $\geq \tau_{\text{reach}}$ and lift $\geq \tau_{\text{lift}}$ and sample_size $\geq \tau_n$, (3) computes removal effects for candidate states, (4) ranks candidates by removal effect, (5) emits findings with predicate `is_activation_driver_for` for top-ranked candidates.

6.2.2 Drop-Off Detector

The drop-off detector identifies states and edges with disproportionately high rates of journey termination without conversion. States with exit_probability $\geq \tau_{\text{exit}}$ and reach_rate $\geq \tau_{\text{reach}}$ are emitted as findings with predicate `is_dropoff_point_for`. Clusters of consecutive drop-off states are aggregated into a single `dropoff_cluster` finding.

6.2.3 Behavioral Regression Detector

The regression detector compares current snapshot metrics to a baseline and identifies statistically significant negative changes using the temporal change detection framework from Section 5. Findings are emitted with predicates `regressed_after` (for release-linked changes) or `changed_after` (for unlinked changes).

6.2.4 Segment Divergence Detector

The segment divergence detector compares behavioral graph metrics across two or more user segments and identifies states or paths that exhibit significantly different conversion rates, reach rates, or transition probabilities. For segment pair (S_1, S_2), it computes the Jensen-Shannon divergence of the transition probability distributions, a symmetric variant that does not require a canonical ordering of the two segments.

Findings are emitted with predicates `more_common_in` and `less_common_in` for reach rate differences, and `diverges_from` for overall distribution divergence.

6.2.5 Repeated Visit Detector

The repeated visit detector identifies states visited multiple times within a single journey, indicating user confusion, friction, or a broken flow. A state s is flagged when the mean

visit count among journeys containing s exceeds a threshold τ_{loop}. Note that this detects unexpectedly high re-visitation frequency; detection of full cyclic paths (graph cycles) requires path-level tracking and is left to future work.

6.2.6 Path Quality Detector

The path quality detector evaluates materialized paths on a composite quality score:

$$Q_{\text{path}} = \text{conversion\_rate} \times \frac{1}{\text{avg\_duration}} \times \frac{1}{\text{path\_length}} \tag{12}$$

High-quality paths are surfaced as findings with predicate `is_fast_path_to`. Low-quality paths with high occurrence rates are flagged as candidate optimization targets. Note that average duration and path length are positively correlated; the multiplicative formulation jointly penalizes both dimensions. This is intentional for the present setting, where long and slow paths are doubly undesirable; however, practitioners may elect to drop one factor if the correlation is strong in a given dataset.

6.3 Detector Composition and Ordering

Detectors run sequentially after each snapshot build. The orchestrator manages dependencies via a directed acyclic graph of detector jobs. Within each job, detectors are independently parallelizable.

7 Grounded Language Layer

7.1 Architecture

The GLL translates structured BKG findings and facts into natural language insights. The key architectural constraint is that the LLM must not perform any numerical computation, must not assert any claim not present in the fact bundle, and must not use causal language where only association is supported.

The GLL implements a three-stage pipeline: (i) fact bundle construction, (ii) grounding validation, and (iii) narrative generation.

7.2 Fact Bundle Construction

For a given finding F, the fact bundle $B(F)$ is constructed by retrieving from the BKG all facts that: (a) have `relatedEntityIds` intersecting the finding's entity set, (b) are valid within the finding's time window, and (c) exceed the minimum confidence threshold. Bundle construction uses a bounded-depth graph traversal (one-hop in the current implementation). This produces a structured context document such as:

```
Finding: activation_driver
State: "import_data"
Predicate: is_activation_driver_for
Object: outcome:converted
Evidence:
  - reach_rate: 0.31
  - P(converted | reached): 0.58
  - P(converted | not reached): 0.18
  - lift: 3.22
  - removal_effect: 0.12
  - sample_size: 4,201
  - confidence: High
Supporting facts:
  - "import_data" commonly_precedes "invite_teammate" (p=0.41)
  - "import_data" is_dropoff_point_for exit (p_dropoff=0.22)
```

7.3 Grounding Validation

Before generation, a grounding validator checks that all numerical values in the fact bundle are self-consistent (e.g., $P(\text{converted} \mid \text{reached})$ and $P(\text{converted} \mid \neg\,\text{reached})$ are within the valid probability range, lift is consistent with both probabilities, sample size meets the minimum threshold). Bundles failing validation are not passed to the LLM.

7.4 Narrative Generation

The validated fact bundle is passed to the LLM with a system prompt that enforces the following constraints: (i) report all numerical values exactly as provided in the fact bundle, (ii) use associative language (*associated with*, *more likely to*, *suggests*) rather than causal language (*causes*, *drives*, *leads to*), (iii) mention confidence and sample size where material, (iv) conclude with a concrete, actionable recommendation.

The resulting narrative is stored alongside the finding in the insight feed, with a provenance pointer back to the fact bundle and the LLM generation parameters.

7.5 Agent Query Interface

In addition to push-based insight delivery, the GLL supports a pull-based query interface. The query is processed by a structured planner that: (1) identifies the relevant journey definition and time window, (2) retrieves candidate facts from the BKG using semantic search over fact summaries, (3) constructs a bounded fact bundle from the retrieved facts, (4) generates a response constrained to the fact bundle. This architecture is analogous to GraphRAG applied to a dynamically constructed behavioral knowledge graph.

8 Evaluation Framework and Open Measurement Challenges

8.1 Challenges Unique to Behavioral Intelligence Systems

Evaluating behavioral intelligence systems presents challenges not present in standard supervised learning settings. There is no ground truth for the set of insights that should be generated from a given event stream. Evaluation must operate across multiple dimensions: computational correctness, statistical validity, and insight quality.

The Markov chain computations underlying Figures 2 and 4 and Table 1 are reproduced by the public simulation scripts at `https://github.com/journium/journium-research`. Figure 3 shows empirical conditional conversion rates $P(\text{converted} \mid \text{reached}(s))$ from synthetic trajectory data; these are distinct from, but consistent with, the absorption probabilities B computed by the scripts (see Section 5). All figures serve as worked examples of formal correctness, not as empirical evaluation of a deployed system.

8.2 Correctness Metrics

Computational correctness is evaluated through: (i) deterministic consistency—for identical input snapshots, the system must produce identical outputs; (ii) traceability—for a random sample of facts, numerical values in the evidence payload must match recomputed values from the raw snapshot data. A target of $\geq 99\%$ traceability is achievable with deterministic computation.

For the GLL, faithfulness is evaluated by extracting verifiable claims from the narrative and checking each against the fact bundle, analogous to the claim-verification methodology of InsightBench [13].

8.3 Insight Quality Metrics

Insight quality is evaluated along three dimensions: precision (are surfaced insights genuinely interesting?), recall (are important behavioral phenomena detected?), and actionability (do insights lead to product actions?). Initial evaluation targets are: activation driver detector precision $\geq 70\%$, regression detector false positive rate $\leq 15\%$, and narrative summaries rated 'actionable' by $\geq 75\%$ of evaluators.

8.4 Latency and Throughput

Core BGE snapshot builds should complete within 1 hour for projects with up to 10M events per day. Detector pipeline execution should complete within 24 hours of the analysis window closing. GLL narrative generation should complete within seconds per finding.

8.5 Comparison with Pull-Based Baselines

BIP is compared against a pull-based baseline on a set of evaluation tasks: identifying the top activation driver, identifying the most significant behavioral regression in a post-release period, and identifying the primary segment divergence. We measure detection rate, time-to-insight, and precision.

9 Discussion

9.1 Design Principles

BIP is built on four design principles that merit explicit discussion.

Grounded-first computation. Every insight in the feed must be traceable to a deterministic computation. The LLM is a linguistic interface, not a reasoning engine. The cost of hallucinated behavioral insights in a production analytics tool is high: misallocated engineering effort, incorrect product decisions, and eroded trust. Grounding is non-negotiable.

Semantic abstraction hierarchy. The three-level state model (raw, semantic, lifecycle) balances expressiveness and tractability. Without semantic abstraction, the graph has too many nodes for meaningful insights; with too much abstraction, the graph loses the specificity needed to guide product decisions.

Time-aware by default. All metrics, facts, and insights are snapshot-scoped. This enables temporal comparison, release-linked change detection, and trend analysis. The decision to make snapshots immutable prioritizes auditability over storage efficiency.

AI on top of truth. The GLL architecture enforces that language generation is strictly downstream of verified computation. The fact schema must carry sufficient numerical detail to enable full regeneration of a faithful narrative, without the LLM needing to infer or compute [12].

9.2 Limitations and Open Problems

- **Causal inference.** Removal effects and lift scores are not causal effects. Identifying which behavioral patterns causally drive activation or conversion requires controlled experimentation or instrumental variable methods beyond the scope of the current architecture.
- **Non-stationarity.** The Markov assumption and the stationarity assumption are known to be violated in practice. Addressing non-stationarity through time-varying Markov models or mixture models [4] is a priority for future work.
- **Cross-journey dependencies.** The journey model treats each journey instance as independent, ignoring network effects, referral chains, and account-level interactions.
- **Learned representations.** The current architecture uses shallow, hand-crafted features. Deep learned representations of event sequences could capture higher-order patterns not visible to the Markov model.
- **Evaluation of interestingness.** The interestingness score defined in Section 5 is a principled approximation, but there is no ground truth for what makes an insight interesting. Developing automated proxies that correlate with human judgment is an open research problem.
- **Privacy and data minimization.** State definitions must be constrained to operate on an allowlist of safe properties, and evidence payloads must not store raw user attributes. Differential privacy techniques for behavioral graph statistics are an important direction.

9.3 Broader Impact

By automating the generation of behavioral insights, BIP has the potential to democratize product intelligence—making sophisticated behavioral analysis accessible to product teams that lack dedicated data science resources. The system is designed to assist and augment human judgment, not replace it; all insights include the evidence and confidence information needed for informed human decision-making.

10 Conclusion

We have presented the Behavioral Intelligence Platform (BIP), a formal architecture for transforming raw product event streams into automatically generated, evidence-backed narrative insights. BIP introduces four tightly integrated layers: a Normalization and State Derivation stage that standardizes raw events and derives a multi-level semantic state hierarchy; a Behavioral Graph Engine that models user journeys as absorbing Markov chains; a Behavioral Knowledge Graph and Detector System that reify graph outputs into a typed triple-store of grounded facts and autonomously identify behavioral phenomena; and a Grounded Language Layer that constrains LLM-generated narratives to verified facts from the BKG.

The architecture is grounded in a formal treatment of the Behavioral Intelligence Problem, with a well-defined taxonomy of detectors, a principled interestingness scoring framework for insight prioritization, and a confidence model that enables calibrated communication of statistical uncertainty. The core design principles—grounded-first computation, semantic abstraction, time-awareness, and AI on top of truth—reflect lessons from adjacent fields including process mining, automated data exploration, knowledge graphs, and retrieval-augmented generation.

BIP represents a step toward analytics systems that take epistemic initiative: systems that do not wait to be asked, but continuously monitor, detect, and narrate the behavioral landscape of a product. The fundamental challenges of causality, non-stationarity, and evaluation of interestingness remain open, and we look forward to collaborative research progress on these problems.

References

[1] Gaël Bernard and Periklis Andritsos. CJM-Miner: Mining customer journey models from customer behavioral data. In *Proceedings of the 20th International Conference on Extending Database Technology (EDBT)*, 2017.

[2] Gartner, Inc. Gartner identifies top 10 data and analytics technology trends for 2019. Press release, Gartner, Inc., 2019.

[3] Gartner, Inc. Gartner predicts 75% of analytics content to use GenAI for enhanced contextual intelligence by 2027. Press release, Gartner, Inc., 2025.

[4] Marius Harbich, Gaël Bernard, Pierre Berkes, Benoît Garbinato, and Periklis Andritsos. Discovering customer journey maps using a mixture of markov models. In *International Symposium on Data-Driven Process Discovery and Analysis*, 2017.

[5] Xinyi He, Mengyu Zhou, et al. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, 2024.

[6] John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. Van Nostrand, Princeton, NJ, 1960.

[7] Po-Ming Law, Alex Endert, and John Stasko. Characterizing automated data insights. In *IEEE Visualization Conference (VIS)*, 2020.

[8] Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *arXiv preprint arXiv:2410.20724*, 2024.

[9] Pingchuan Ma, Rui Ding, Shi Han, and Dongmei Zhang. MetaInsight: Automatic discovery of structured knowledge for exploratory data analysis. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD)*. ACM, 2021.

[10] Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. Demonstration of InsightPilot: An LLM-empowered automated data exploration system. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2023.

[11] Aamod Manatkar et al. QUIS: Question-guided insights generation for automated exploratory data analysis. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP)*, 2024.

[12] Shirui Pan et al. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[13] Gaurav Sahu, Abhay Puri, Juan Rodriguez, Amirhossein Abaskohi, et al. InsightBench: Evaluating business analytics agents through multi-step insight generation. *arXiv preprint arXiv:2407.06423*, 2024. Accepted at ICLR 2025.

[14] Abraham Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD)*, 1995.

[15] Wil M. P. van der Aalst. *Process Mining: Data Science in Action*. Springer, Berlin, Heidelberg, 2nd edition, 2016.

[16] Hanwen Wang et al. Customer behaviour hidden Markov model. *Mathematics*, 10(8):1230, 2022.

[17] Kevin Xu, Xiao Ma, and Dongmei Zhang. DataShot: Automatic generation of fact sheets from tabular data. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[18] Qian Zhang et al. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*, 2025.