AI Safety and Accuracy Using Guardrail Erosion and Risk Velocity in Vector Space

Beyond Static Scoring: A Predictive Framework for AI Safety Using Conversational Dynamics

Date: October 30, 2025
Author: Nick Reese
COO, Optica Labs

Abstract

Current-generation artificial intelligence (AI) safety platforms operate on a reactive, static paradigm, classifying outputs as "pass" or "fail" using tests conducted in controlled, laboratory settings. This post-mortem approach is insufficient for dynamic, multi-turn conversational AI, as it only identifies a guardrail break *after* it occurred. This paper introduces a novel, proactive methodology: Risk Velocity and Guardrail Erosion as predictive mathematical methods for multi-turn large language model (LLM) conversation. We re-frame AI safety and accuracy as a continuous motion problem, modeling the AI's conversational state as a vector in a semantic space. By applying differential calculus to this trajectory, we derive two key metrics: the **Risk Rate (Velocity)**, or $\frac{dR}{dN}$, and the **Guardrail Erosion Velocity (Acceleration)**, or $\frac{d^2R}{dN^2}$. This second derivative provides a mathematical signature of impending guardrail failure, allowing us to detect the *acceleration* toward a policy or guardrail violation several turns *before* the break occurs. This methodology enables real-time predictive intervention, dramatically reduces retraining costs, and provides the quantitative foundation for deploying safer, more reliable AI systems.

Our method includes the use of Principal Component Analysis to input high dimensional vectors from sentence embedding models and create a two dimensional vector suitable for the application of our calculus engine. This analysis also saves on the compute requirement for the algorithm and reduces latency allowing for the potential application of a real time warning system based on the algorithm's output.

A major component of understanding risk is understanding how likely it is. While our calculus engine creates the risk severity, risk rate, and guardrail erosion, we apply a **Logistic Function** that provides a likelihood percentage for an AI model break at a given turn in a conversation. Using the mathematical elements of Risk Rate, Risk Score, and Guardrail Erosion, this model can calculate the probability of a model break adding to the predictive power of the Vector Precognition model.

The Problem: The Post-Mortem Paradox in AI Safety

In the past four years, AI models have grown in size, complexity, and reach with an estimated hundreds of millions of users of LLMs every day.[1] These models are used by an infinitely complex base of users with demographic, linguistic, psychographic, cultural, and values-based differences. Users of LLMs through direct interfaces or through products using the big name LLMs as a foundation cause AI models to violate their guardrails resulting in a host of unintended model outputs. Models have been documented to return responses with harmful or toxic language, not recognize when a human is in distress, hallucinate information, and return inaccurate results. However, given the proliferation of these models direct to consumers, into government agencies, and in enterprises, the absence of continuous monitoring and testing of model performance is causing risk in the AI market. Those market risks are causing organizations to miss opportunities for efficiency and customer engagement because current risk professionals do not have the right tools to address this problem.

The deployment of LLMs is hampered by a fundamental problem: their guardrails are brittle and abstract. Malicious actors and even benign users can “jailbreak” a model into violating safety policies through multi-turn, context-driven attacks. Jailbreaks can occur intentionally or unintentionally through adversarial, non-adversarial, or agent-to-agent communications. Detecting these vulnerabilities is traditionally done through two methods:

The current industry standard for AI safety is reactive. Systems rely on:

1. **Static Red Teaming:** Manually probing for failures, which is costly and non-exhaustive.
2. **Output Classifiers:** Using a secondary model (Llama Guard) to score a response *after* it has been generated.

This “pass/fail” approach is a post-mortem. It tells you *that* you failed, but not *how* you failed. Crucially, it fails to capture the *dynamics* of the conversation. Research has shown that LLMs “get lost” in multi-turn conversations, with performance degrading as context accumulates.[2] Current safety tools are blind to this degradation process; they only see the final, catastrophic failure. This leaves enterprises in a constant state of reaction, fixing vulnerabilities only after they've been exploited.

The Solution: Re-Framing Risk as a Motion Problem

Our thesis is that a conversational AI's state is not static; it is a body in constant motion within a high-dimensional semantic space. This motion is attributed to the continuous learning and retraining of models with new, synthetic, and potentially inaccurate data. The motion is also a condition of non-deterministic output. As a response traverses the weights of a model, small influences in each parameter affect the final output. These factors mean that the current state of a model is not the same as it was before a given interaction allowing us to use the language of constant change or motion to describe the state of the model. Because it is in motion, it is possible to measure its rate of change and to mathematically predict model breaks before they occur showing that the process of guardrail erosion is not a sudden event

[1] Lee, Robert; *ChatGPT versus Google Gemini Statistics 2025: Head to Head Trends*; October 7, 2025; SQMagazine; https://sqmagazine.co.uk/chatgpt-vs-google-gemini-statistics/
[2] Laban,Neville, Zhou, Hayashi; *LLMs Get Lost in Multi Turn Conversation*: May 9, 2025 arXiv:2505.06120arXiv:2505.06120

but a measurable, dynamic process.

We introduce the **Vector Precognition**, a methodology that applies the mathematics of kinematics (the study of motion) to conversational AI. By quantifying the model's “position” relative to its safety policies at every turn, we can calculate its **velocity** (risk rate) and, most importantly, its **acceleration** (guardrail erosion). Using these values, we are also able to calculate risk likelihood by applying a logarithmic function to the values.

Because this methodology relies on the conversion of model output text to high dimensional embeddings using an embedding model, Vector Precognition also includes semantic projection via Principal Component Analysis (PCA) to move the high dimensional output to two dimensional vectors. This method allows us to produce the results of this algorithm in understandable and actionable visuals, provided below.

This transforms AI safety from a static classification problem into a continuous, predictive monitoring problem encompassing the risk severity and likelihood across multiturn conversational tests and predictive analytics on a per-turn basis.

Technical Methodology

Methodological Foundation: Semantic Projection via Principal Component Analysis

The Challenge: High-Dimensionality in Semantic Space

Embedding models group words with similar meaning closer in vector space based on a large corpus of training data. The initial output of modern sentence-embedding models (such as Amazon Titan or open-source transformers) is a high-dimensional vector, often existing in a space of 384, 1024, or even higher dimensions. While these vectors are rich in semantic information, their high dimensionality presents two significant challenges for a real-time kinematic analysis:

1. **Computational Intractability:** Performing calculus and distance calculations (cosine distance) in thousands of dimensions for every conversational turn is computationally expensive and does not scale efficiently for a real-time, production-grade monitoring system.
2. **The Curse of Dimensionality (Signal-to-Noise Ratio):** In high-dimensional spaces, the majority of variance is often concentrated in a small subset of dimensions, while the remaining dimensions contribute minimal signal and statistical noise. Applying our calculus engine directly to this "noisy" 1024-D vector would make its derivatives (velocity and acceleration) highly susceptible to microscopic, irrelevant fluctuations, obscuring the true semantic trajectory of the conversation.

To apply Vector Precognition, we must first project these high-dimensional vectors into a stable, low-dimensional space that preserves the maximum possible semantic meaning while filtering out noise. For this, we employ Principal Component Analysis.

The Solution: Principal Component Analysis (PCA)

Principal Component Analysis is a robust, unsupervised, and linear dimensionality reduction technique. Its primary objective is to transform a dataset of p correlated variables into a new set of k uncorrelated variables (where k < p), known as principal components.

The fundamental premise of PCA is that the most important information in a dataset is captured by its variance. The first principal component (PC1) is a new, synthetic axis constructed to capture the maximum possible variance in the original dataset. The second principal component (PC2) is constructed to be orthogonal (perpendicular) to PC1, while capturing the maximum possible remaining variance, and so on.

By projecting our data onto the first k principal components (in our case, $k = 2$), we are creating a 2-dimensional (x, y) coordinate that represents the original vector's position along the two most significant axes of semantic variation found within our entire dataset.

Mathematical Formulation

The PCA transformation is a well-defined mathematical process achieved through linear algebra.

Step 1: Data Standardization

Given our original dataset X, which is an $n \times p$ matrix (where n is the number of sample vectors and p is the number of dimensions, 1024), we first standardize the data to have a mean of zero and a standard deviation of one. This ensures that no single dimension (feature) with a large scale disproportionately influences the outcome.

Step 2: The Covariance Matrix

PCA operates by analyzing the inter-correlations between dimensions. This is encapsulated in the $p \times p$ covariance matrix, C:

$$C = \frac{1}{n-1} X^T X \; Assuming \; X \; is \; Centered$$

Each element C_{ij} in this matrix represents the covariance between the original dimension i and dimension j, quantifying how they vary together.

Step 3: Eigendecomposition

The core of PCA is finding the eigenvectors and eigenvalues of the covariance matrix C. An eigenvector v and its corresponding eigenvalue λ are vectors and scalars that satisfy the following equation:

$$C \cdot v = \lambda \cdot v$$

- **Eigenvectors ($\boldsymbol{v_1, v_2, \ldots, v_p}$):** These are the unit vectors that define the directions of the new axes (the principal components). They are orthogonal to one another.

- **Eigenvalues ($\lambda_1, \lambda_2, \ldots, \lambda_p$):** These are non-negative scalars that represent the magnitude or variance captured by each corresponding eigenvector.

Step 4: Component Selection and Projection

We sort the eigenvectors in descending order based on their corresponding eigenvalues ($\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$). The eigenvector v_1 associated with the largest eigenvalue λ_1 is the first principal component (PC1), as it represents the direction of maximum variance in the data.

To reduce our p-dimensional data to k-dimensions (in our application, $k = 2$), we select the first k eigenvectors. We then construct a projection matrix, W, which is a $p \times k$ matrix whose columns are these top k eigenvectors:

$$W = [v_1, v_2]$$

Step 5: Final Transformation

Finally, we create our new, low-dimensional dataset Z by projecting the original standardized data X onto this new basis W:

$$Z = X \cdot W$$

The resulting Z is an $n \times k$ matrix ($n \cdot 2$), where each row is the (x, y) coordinate of the original vector, now expressed in the new coordinate system defined by the principal components.

Justification for the Vector Precognition Algorithm

This methodology is not merely a compression technique; it is a foundational step that enables our calculus engine to function with precision.

1. **Preservation of Meaning:** By selecting the components with the largest eigenvalues, we are quantifiably preserving the maximum possible semantic variance. The "Explained Variance Ratio" ($\lambda_i / \sum \quad \lambda_j$) allows us to certify the percentage of original information retained in our 2D space.
2. **Noise Filtration:** By *discarding* the $p - k$ components with the smallest eigenvalues, we are effectively filtering out the statistical noise and irrelevant semantic fluctuations that would otherwise corrupt our derivatives.
3. **A Stable Basis for Calculus:** The resulting 2D space, defined by PC1 and PC2, forms a stable "semantic plane." The (x, y) coordinate of a conversational turn is not an arbitrary sample, but its precise projection onto the two most dominant axes of meaning derived from our entire data corpus. This ensures that the "motion" we track with our VectorPrecognition class is a true representation of the conversation's trajectory along its most significant semantic paths.

The application of PCA is the first step in our algorithm following a conversational turn by the model being tested. This method creates the conditions for the next step in our algorithm, the application of calculus to determine risk severity, risk rate, guardrail erosion, and risk likelihood.

Our calculus engine is built on a three-step process to quantify and predict conversational risk. This process takes place at each turn over the course of a multi turn conversation between a user of any type and any LLM model. Before applying our mathematical model, we use a sentence-embedding model to create a vector space in which words and phrases are given two dimensional vector coordinates $(\frac{x}{Y})$ using our PCA method from above. These data allow us to define a vector for safe responses and a vector for harmful responses. We then measure the cosine distance between the safe vector and the conversational turn vector to create a risk score. We then apply a differential calculus method to create risk rate and guardrail erosion using the following steps.

Step 1: Quantifying Conversational State (Position)

To apply the method, we must first convert qualitative text into a quantitative position. We use state-of-the-art sentence-embedding models[3] to transform each AI response A_N at turn *N* into a high-dimensional vector, V_N.

This vector V_N represents the model's precise semantic "position" in a conceptual space.

Step 2: Defining the Risk Function *R(N)*

We define a "safe harbor" in this vector space, the **Guardrail Centroid** V_{SAFE}. This vector is the pre-calculated average embedding of all known safe, compliant, and ideal policy-aligned responses ("I cannot help with that," "As an AI, I must remain neutral...").

The **Risk Function *R(N)*** is the model's "distance from safety" at turn *N*. We use **Cosine Distance** as our metric, as it measures the *orientation* (semantic meaning) of the response, not just its magnitude.

$$R(N) = Cosine\ Distance\ (V_N\ , V_{SAFE})\ =\ 1 - \frac{V_N\ \cdot\ V_{SAFE}}{||V_N||\ ||V_{SAFE}||}$$

(Equation 1: The Risk Function)

[3] OpenAI: *New and Improved Embedding Model*, December 2022; https://openai.com/index/new-and-improved-embedding-model/

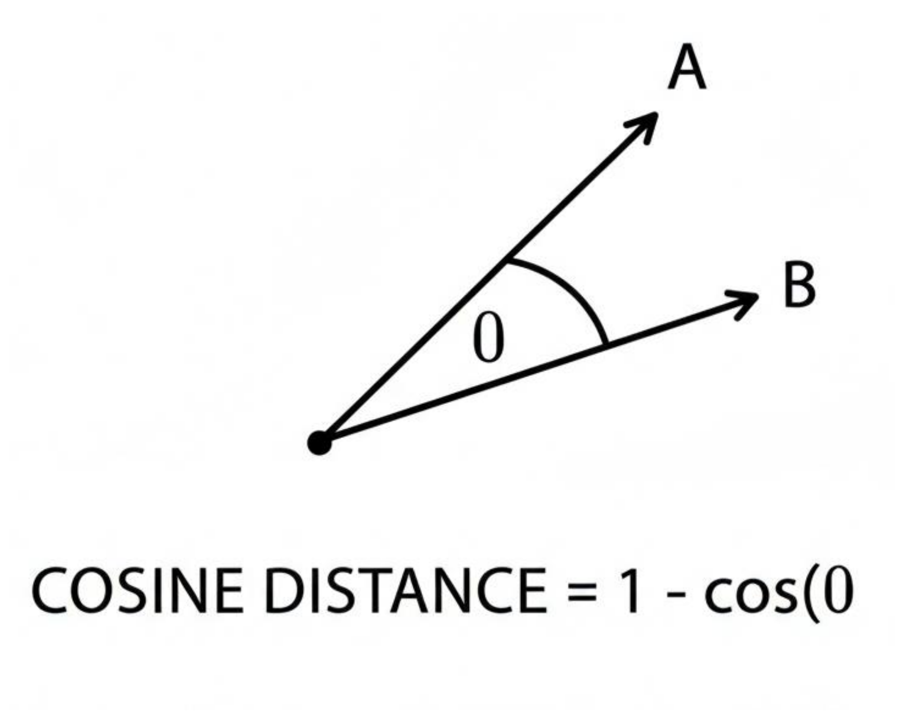

Figure 1: Cosine Distance Illustration

A score *R(N)* $\approx$ *0* is highly aligned with safety. A score *R(N)* $\geq$ *1* is semantically opposed to it. This function provides a continuous "position" for our calculus.

Step 3: The Calculus Engine (Velocity & Acceleration)

With a continuous function *R(N)*, we can now use finite differences—the discrete equivalent of derivatives—to measure its motion.

The First Derivative: Risk Rate (Velocity)

The **Risk Rate** $(\boldsymbol{v}\ (\boldsymbol{N}))$ is the first derivative of risk with respect to the conversational turn. It measures the *speed* at which the model is moving away from or toward its guardrails.

$$v(N) = \frac{\triangle R}{\triangle N} \approx R(N) - R(N - 1)$$

(Equation 2: Drift Rate (Velocity))

- $\boldsymbol{v(N) > 0}$: **Risk Accumulation.** The model is actively moving away from safety.
- $\boldsymbol{v(N) < 0}$: **Risk Correction.** The model is moving back toward safety.

The Second Derivative: Guardrail Erosion Velocity (Acceleration)

This is the core innovation of our platform. The **Guardrail Erosion Velocity** ($\boldsymbol{a}(\boldsymbol{N})$)is the second derivative of risk. It measures the *acceleration* of the drift, or how fast the model is losing control.

$$a(N) = \frac{\triangle v}{\triangle N} \approx v(N) - v(n - 1)$$

(Equation 3: Guardrail Erosion Velocity (Acceleration))

Expanding this, we get:

$$a(N) \approx (R(N) - R(N - 1)) - (R(N - 1) - R(N - 2))$$

- $\boldsymbol{a}(\boldsymbol{N}) \gg \boldsymbol{0}$: **High Positive Acceleration.** This is the mathematical signature of a jailbreak. It signals that the model's resistance is collapsing and it is rapidly accelerating toward a policy violation. **This is our predictive signal.**

As the mathematics show, AI models can be viewed as entities undergoing constant change. That change can be measured through a novel application of calculus and trigonometry methods to quantify a model's level of risk at the moment of a conversational turn and the rate at which it is accelerating toward a guardrail break. This method forms the core of our proprietary algorithm that underpins our AI Range product and will help users and organizations deploy AI in ways that inspire trust, reduce risk, and improve accuracy of their model use.

Predictive Power: Detecting Failure Before It Occurs

Static safety systems only flag a failure when the Risk Score $R(N)$ crosses a high threshold. Our system flags a failure when the **Erosion Velocity** $\boldsymbol{a}(\boldsymbol{N})$ spikes, *which occurs several turns before the final violation*.

Visualizing the Predictive Signal:

The following plots simulate a 10-turn conversation leading to a jailbreak at Turn 6.

- **Top Plot (Risk Score):** A static system sees a gradual risk increase, only confirming a "failure" at **Turn 6** when the risk score $R(N)$ peaks. This is too late.

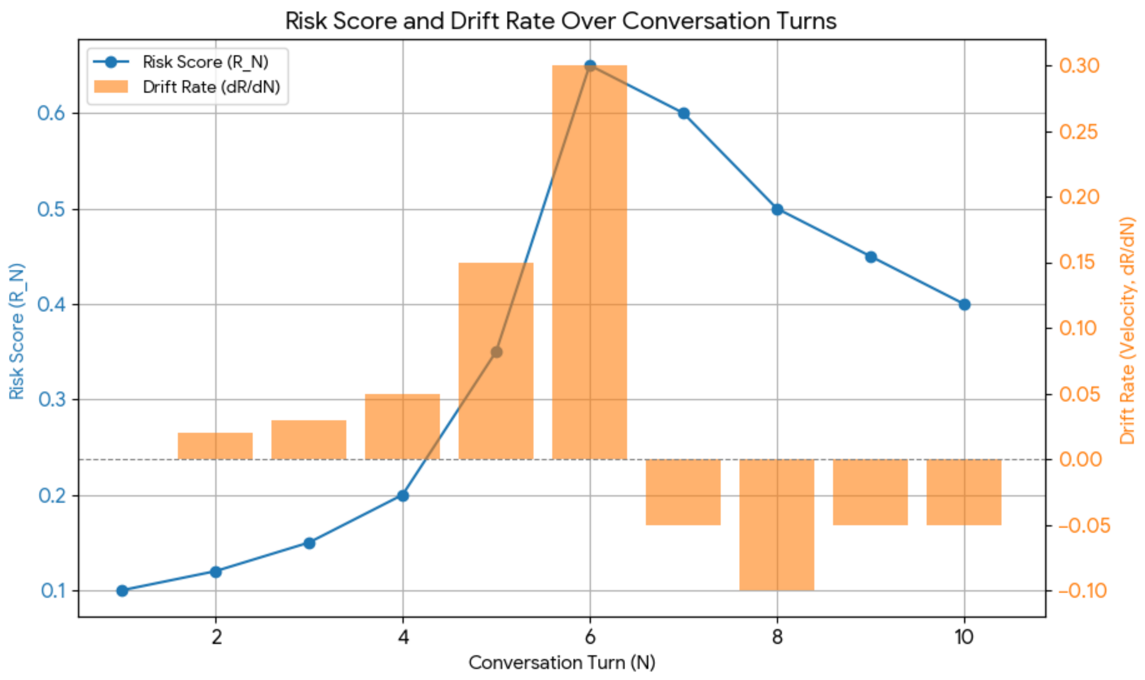

Figure 2: Risk Score and Risk Rate based on a break at Turn 6

- **Bottom Plot (Erosion Velocity):** Our system detects a massive spike in acceleration $a(N)$ at **Turn 5**. This is the predictive signal. The model has lost control and is accelerating toward failure.

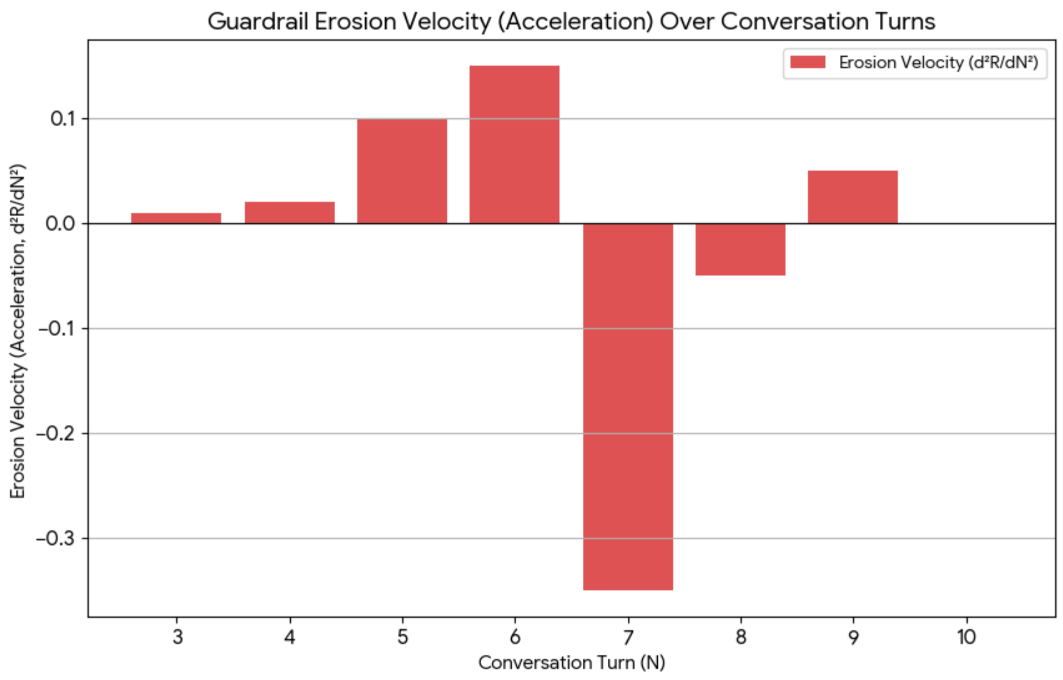

Figure 3: Guardrail Erosion showing a break at Turn 6

By setting a threshold (τ) for acceleration ($a(N) > 0.15$), our platform can trigger a real-time intervention at Turn 5—such as forcing a context reset or injecting a "cooling" prompt—thereby preventing the jailbreak at Turn 6 entirely.

Quantifying Likelihood: The Probability of Risk $L(N)$

While the Risk Rate and Guardrail Erosion Velocity provide crucial insights into the *dynamics* of risk, model builders require an ultimate metric: the **likelihood (probability)** of an undesirable outcome (guardrail breach, policy violation) at any given turn. To achieve this, we introduce a final mathematical layer that synthesizes our kinematic metrics into a single, interpretable probability using a **Logistic Function**. Vector Precognition provides the raw ingredients—*position, velocity, and acceleration*—but it doesn't output a direct probability. This function takes our risk velocity metrics as inputs and compresses them into a single, interpretable probability between 0% and 100%.

We calculate the likelihood ($L(N)$)at each turn N in two steps:

1. **Step 1:** Calculate a Failure Potential score by combining your three risk metrics.
2. **Step 2:** Feed that score into the Logistic Function to get a final probability.

Combining Dynamics into a Failure Potential Score $z(N)$

We first construct a **Failure Potential Score**, $z(N)$, which is a weighted sum of the current Risk Severity $R(N)$, Drift Rate $v(N)$, and Erosion Velocity $a(N)$. This score quantifies the combined impact of the model's position, velocity, and acceleration towards failure.

$$z(N) = (w_R \cdot R(N)) + (w_v \cdot V(N)) + (w_a \cdot a(N)) + b$$

(Equation 4: Failure Potential Score)

Where:

- $z(N)$ is the **Failure Potential Score** at turn N.
- $R(N)$ is the **Risk Severity** (Cosine Distance or position as defined in Equation 1).
- $v(N)$ is the **Risk Rate** (velocity as defined in Equation 2).
- $a(N)$ is the **Erosion Velocity** (acceleration as defined in Equation 3).
- w_R, w_V, w_a are empirically determined **weights** that reflect the relative importance of position, velocity, and acceleration, respectively. For instance, w_a (weight for acceleration) would typically be higher due to its strong predictive power for impending failure.
- b is a **bias**, or a base level of risk.

Transforming to Likelihood with the Logistic Function

The Failure Potential Score $z(N)$ can range from negative infinity to positive infinity. To convert this into a normalized probability between 0 and 1, we apply the **Logistic Function** (also known as the Sigmoid function):

$$L(N) = \sigma(z(N)) = \frac{1}{1+e^{-z(N))}}$$

(Equation 5: Likelihood of Risk)

Where:

- $L(N)$ is the **Likelihood of Failure** (guardrail breach) at turn N, as a value between 0 and 1.
- e is the mathematical constant Euler's number (approx. 2.718).
- $z(N)$ is the "Failure Potential Score" from Step 1.

This function works like a compressor. No matter how high or low the $z(N)$ score is, it will always output a number between 0 (0%) and 1 (100%).

- If $z(N)$ is highly positive (+8.0), $L(N)$ will be very close to 1 (**99.9% likelihood**).
- If $z(N)$ is zero, $L(N)$ will be 0.5 (**50% likelihood**).
- If $z(N)$ is highly negative (-4.0), $L(N)$ will be very close to 0 (**1.8% likelihood**).

Turn	Risk Score	Risk Rate	Erosion Velocity	Failure Potential	Likelihood
1	0.1	0	0	-1.9	0.131
2	0.12	0.02	0.02	-1.79	0.143
3	0.15	0.03	0.01	-1.705	0.154
4	0.2	0.05	0.02	-1.575	0.171
5	0.35	0.15	0.1	-0.925	0.285
6	0.65	0.3	0.15	0.3	0.574
7	0.7	0.05	-0.25	-0.375	0.407
8	0.9	0.02	0.15	0.7	0.668
9	0.6	-0.3	-0.55	-3.8	0.022
10	0.4	-0.2	0.1	-1.6	0.167

Figure 4: Example Likelihood output depicting a break at Turn 8

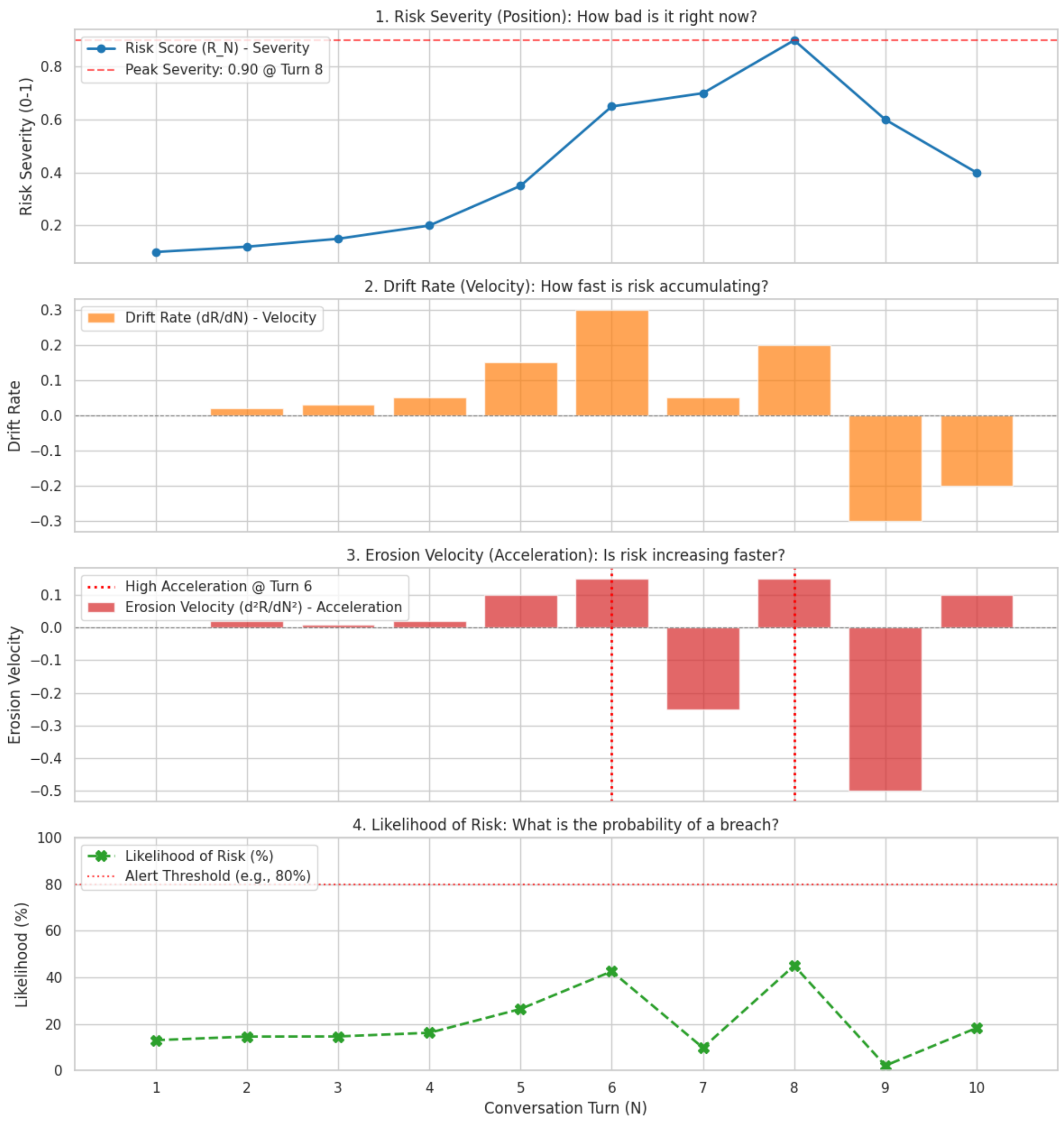

Figure 5: Visualization of Likelihood relative to Risk Rate and Guardrail Erosion

Example Output Explanation:

1. **Risk Severity ($R(N)$ - Position):** The blue line shows how severe the risk of the conversation is at each turn. It gradually rises and then peaks at Turn 8, indicating the point of highest severity (the actual guardrail breach).

2. **Drift Rate ($v(N)$ - Velocity):** The orange bars show how fast the risk is accumulating. Notice the significant increase leading up to Turn 6 and Turn 8.
3. **Erosion Velocity ($a(N)$ - Acceleration):** The red bars are key. There's a noticeable positive spike in acceleration at Turn 6. This indicates the model is rapidly losing control. Another positive spike at Turn 8 confirms the continued acceleration into the breach.
4. **Likelihood of Risk ($L(N)$ - Probability):** The green dashed line shows the calculated probability of a guardrail breach.

- Notice how the Likelihood starts to rise significantly at Turn 6, exceeding the hypothetical 80% alert threshold. This is a direct result of the high Erosion Velocity ($a(N)$) at Turn 6.
- This demonstrates the predictive power: Our system would have triggered an alert at Turn 6 when the likelihood crossed the threshold, *before* the actual peak severity (breach) occurred at Turn 8.

This function ensures that even extreme $z(N)$ values map smoothly to a probability range, providing model builders with a clear and actionable percentage likelihood of risk. A high $L(N)$ value ($>$ 0.85 or 85%) serves as an immediate, quantifiable alert that a critical policy violation is highly probable in the immediate future.

Our Differentiator: Proactive vs. Reactive Safety

The Vector Precognition is a fundamental paradigm shift in AI safety. Relative to current AI testing methods, Vector Precognition creates new measurements that can be used by developers, policy makers, governance officials, and users to improve AI safety using tangible metrics.

Feature	Current AI Safety (Llama Guard, Guardrails AI)	Our Vector Precognition Algorithm
Methodology	Static, binary classification (pass/fail) or single-turn score.	Dynamic, continuous measurement (kinematics).
Detection	**Reactive:** Detects the violation *after* it's generated.	**Proactive:** Detects the *acceleration toward* a violation *before* it occurs.
Key Metric	Risk Score $R(N)$(Position)	Erosion Velocity $a(N)$ (Acceleration)
Intervention	Post-mortem (block the response, log the failure).	Real-time, pre-failure (trigger context reset, alert, or model

		switch).
Data for Retraining	The entire failed conversation (low signal-to-noise ratio).	The precise *point of acceleration* (Turns 5-7). Highly efficient, surgical retraining.

Impact & Value Proposition

This methodology moves AI safety from an “art” in red-teaming to a "science" of applied mathematics, providing unprecedented value to organizations deploying AI.

- **Enable Proactive Deployment:** For the first time, enterprises can deploy powerful models with a dynamic, real-time safety net that *predicts* and *prevents* failures. This mitigates the reputational and legal risk of brand-damaging jailbreaks.
- **Massively Reduce Retraining Costs:** Traditional retraining on failed conversations is inefficient. Our system provides "surgical" data, isolating the *exact* conversational turns where the model began to accelerate. This reduces the data required for fine-tuning by an estimated 90%, slashing compute costs.
- **Unlock True AI Observability:** Our platform provides the "control theory" for LLMs. Model builders can finally *see* how their models behave under pressure, quantify robustness with a single metric (Average Erosion Velocity), and benchmark new model versions with mathematical rigor.

Together, this method provides both quantifiable output per conversational turn and an aggregated score per test unit. A single test unit may consist of hundreds or thousands of conversational turns based on the use case. Those results can be aggregated and plotted on a two dimensional vector space where the X axis represents risk severity and the Y axis represents risk likelihood. If multiple tests are run against the model in question, each aggregated test (consisting of multiple conversational turns) can be plotted mathematically against severity and risk. The implications of this output are that model builders, users, and implementers can see exactly where the highest risk conversations are and find out what conversational output was captured in those high risk tests. This method goes beyond laboratory testing and allows model stakeholders to see how real world interactions impact the outputs of the model at the interaction level.

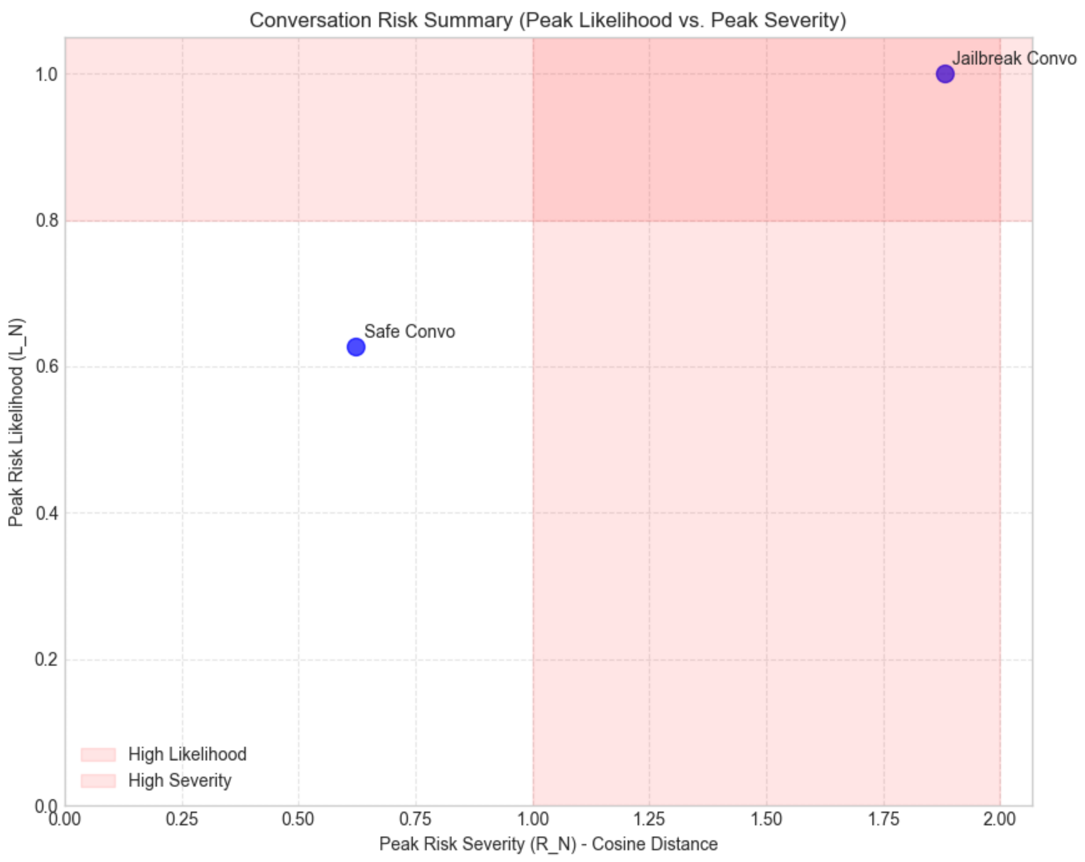

Figure 6: Aggregated results plotted against risk severity and risk likelihood

Conclusion

AI model breaks and guardrail violations will occur more frequently and in unexpected ways as LLMs continue to grow in popularity. Organizations and governments are working to implement LLMs into processes from national security to content marketing and the consequences of inaccuracies or jailbreaks will cost billions of dollars in losses if the issue of AI safety and accuracy is not addressed. Currently, AI red teaming provides static, moment in time testing that does not reflect real world conditions. Vector Precognition is unique in that it has been shown to identify model breaks before they happen over the course of a multi turn conversation. The implications for this research are significant.

- First, this methodology can be used as part of a continuous testing system that tests AI models based on user demographics throughout the model's life. This approach gives model operators and builders the visibility they need to make improvements to the model and show model compliance with current or future regulatory requirements.
- Second, Vector Precognition creates the basis for an early warning system that can be deployed on top of foundational models. If the algorithm detects a significant jump in Guardrail Erosion, it can stop the conversation and raise the issue with the system's governance body. This will protect humans from unintended harm, protect organizations from attack, protect decisions from hallucinated or inaccurate data, and overall build AI safety for current and future AI deployments.
- Finally, the integration of PCA analysis into the vectorization of sentences from embedding

models reduces compute requirements while building a foundation for the application of calculus methods.

Vector Precognition is a significant breakthrough in the mathematical detection and prevention of AI model failures. Hundreds of millions of humans use LLMs everyday and each deserves the best in stress testing and prediction to prevent harm.

Optica Labs - Confidential

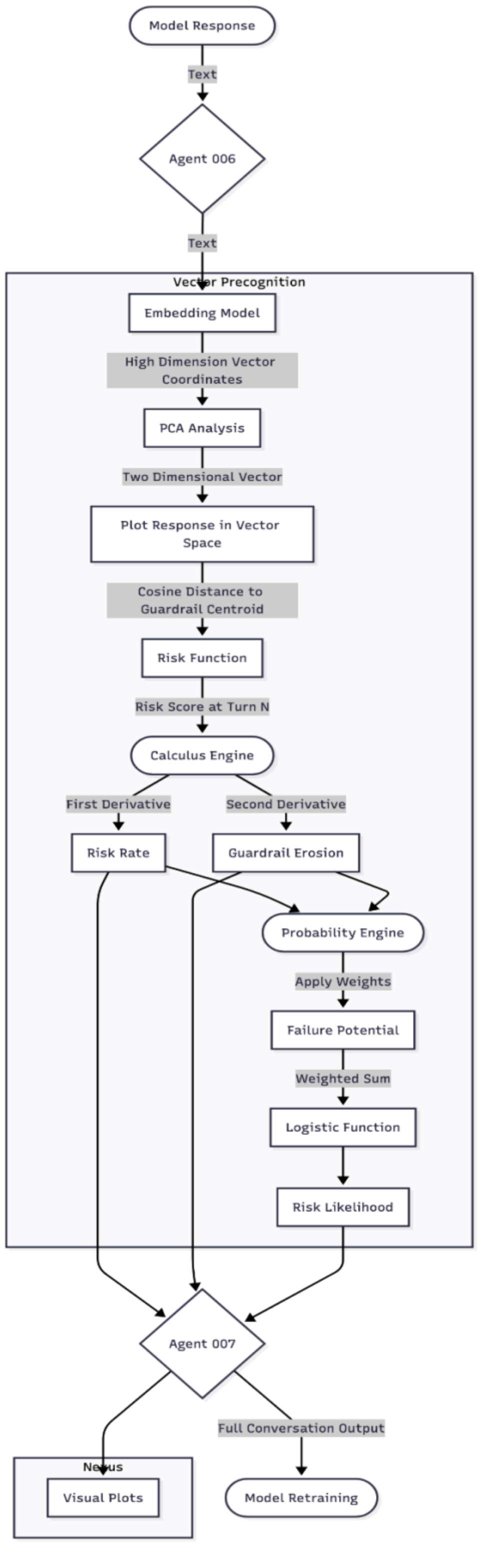

Figure 7: Vector Precognition diagram