

Learning to Detect Shifts in Unstructured Data Streams via Monitoring-by-Betting

Siyu Zhou¹, Jialin Ouyang¹, Fan Xia¹, Adarsh Subbaswamy², Drew Prinster³, Suchi Saria³,
Augusto Garcia-Agundez¹, Sarah Pollet¹, Jinoos Yazdany¹, Patrick Vossler^{1,†}, Jean Feng^{1,†}

¹University of California, San Francisco ²University of Maryland ³Johns Hopkins University

Abstract

Continuous monitoring of generative AI systems requires detecting distribution shifts in the unstructured data they ingest and generate. Existing monitoring methods typically control false alarms by requiring the discrepancy measure used to detect shifts to be *pre-specified*. However, for unstructured data, pre-specification creates a sensitivity–breadth trade-off: a narrowly defined measure is sensitive to the anticipated shift but may miss unexpected changes, whereas a broadly defined measure may detect a wider range of shifts but with reduced sensitivity. Although adaptively hypothesizing the shift direction can overcome this limitation, such data-dependent adaptation generally invalidates conventional false-alarm guarantees. We reconcile adaptivity with statistical validity through testing-by-betting, reframing sequential changepoint detection as a betting game in which the bettor continually updates its hypothesized shift direction, or witness function, as new observations arrive. We introduce *asymptotic e-detectors*, which approximate the pre-change distribution using a calibration set of size m , and prove that they asymptotically control the average run length to a false alarm as m increases. Empirically, calibration sets as small as $m = 30$ provide effective control. We then develop adaptive monitoring strategies ranging from black-box bettors that learn from embeddings to interpretable bettors based on LLM-proposed concepts. Across three real-world datasets, including data from a clinical AI scribe deployment, adaptive monitors detect shifts faster than fixed monitors at the same false-alarm level. Further, when the true shifts are sparse and interpretable, LLM-concept monitors provide the fastest and most interpretable detection, while also guiding root-cause diagnosis for complex shifts.

1 INTRODUCTION

Preventing performance decay in deployed generative AI systems requires monitoring throughout the entire pipeline, including the input distribution, user behavior, and model outputs. Existing monitoring procedures follow the general recipe of pre-specifying a discrepancy measure and tracking shifts with respect to this metric. Pre-specification plays a critical role [46, 63, 65], as it ensures control over false alarms; this is typically measured by Average Run Length under the null hypothesis of there being no changepoint (ARL_0), which quantifies the “lifespan” of a monitor until it must be reset. For instance, LLM-based monitoring methods track scores with respect to a fixed performance rubric [35] or shifts in fixed directions away from the original data distribution [74, 43, 70]. Other methods convert unstructured data into high-dimensional vectors, which can then be monitored for shifts with respect to pre-specified directions [7, 12, 50] or kernel functions [31, 9]. Monitoring metrics should be specified with immense care, as performance of monitoring procedures is highly sensitive to this choice [26]: strong alignment between the pre-specified measure and the future shift direction leads to fast detection, while poor alignment may lead to having little to no detection power.

Nevertheless, pre-specification has key limitations that leave it misaligned with modern AI practice. First, it is typically difficult to pre-specify the monitoring metric, since future shift directions are, by definition, unknown. This forces monitoring teams to choose between concentrating power on detecting shift in particular directions (sensitivity) versus spreading power across many possible directions (breadth). Defining shifts for high-dimensional or unstructured data is particularly difficult given that such data can experience a wide variety of shifts. As such, recent works on retrospective testing have instead sought to *learn* so-called “witness functions” that highlight differences between data distributions by training a classifier [44, 41, 40]. Second, pre-specification prevents the monitoring metric from adapting as information accumulates, whereas

[†]Co-senior authors.

a monitoring team may gain a better understanding of the data and likely shift directions as data accumulates. For instance, a shift may become obvious to a human after a small number of observations [24]. In fact, the rigidity of current statistical monitoring procedures has led monitoring to be viewed as a fundamentally “human” activity [45], in which statistical evidence complements, rather than replaces, human judgment. Finally, with the increasing reasoning capabilities of LLMs, monitoring practitioners are increasingly drawn to using LLMs to assist in monitoring [3], but formal statistical monitoring procedures still lack the ability to incorporate such adaptations on-the-fly.

As such, the goal of this paper is to answer the following question: *can we monitor unstructured data streams by adaptively increasing sensitivity to shifts learned from past data, while still maintaining statistical guarantees over false alarms?* To this end, we look to the Safe Anytime-Valid Inference (SAVI) framework [54], which enables nearly arbitrary adaptations during the testing process by reframing sequential testing as a game of betting against the null hypothesis. This game must satisfy one simple martingale-like property: the game’s returns should be centered so that any bettor—however adaptive and adversarial—cannot gain wealth in expectation under the null hypothesis [59, 30, 52]. A bettor’s current wealth is thus the current level of evidence against the null, and rejection of the null is warranted when the bettor’s wealth exceeds some threshold; this wealth is called an “e-value,” as the analog of the classical p-value.

In sequential changepoint detection, the null hypothesis is that there is no change, i.e., $H_0 : P_t = P \quad \forall t = 1, 2, \dots$, while the alternative is that the data switches to a post-change distribution Q after some changepoint τ . To control ARL_0 , recent work has built on e-values and the SAVI framework to introduce “e-detectors” as a broad class of procedures [62]. Through the betting lens, these methods can be generally viewed as playing the following game:

Changepoint-Detection-Game

At each time t :

1. Based on data up to time $t-1$, the bettor proposes a witness function $\hat{f}_t$, whose values are expected to be large post-change but small pre-change, and a bet scaling factor λ_t .
2. The game’s return at time t is the scaled difference between the value of $\hat{f}_t$ on the new observation X_t and its expectation in the pre-change distribution.

Playing the game is thus quite simple: at each time point, the bettor adaptively hypothesizes when/what changed in the stream and designs a witness func-

tion to detect this. Indeed, humans may actually be quite adept at this game, as a human who is manually monitoring a stream of text or images will naturally hypothesize shift directions (e.g., images become more blurry) and check alignment of incoming observations against these candidate directions.

On the other hand, *setting up the game* is far from straightforward, because the exact expectation of the witness function is unknown in practice; so it is difficult to center the game’s returns to satisfy the martingale requirement and guarantee there is no winning strategy under the null. To our knowledge there are currently no highly adaptive changepoint detection methods for the most basic setting of monitoring a single stream of unstructured data. Instead, prior works have mainly explored variants of this game, such as monitoring paired data streams where the paired structure simplifies the design of game returns [61, 47]. However, one typically does not have access to a second parallel data stream representing the relevant pre-change distribution (e.g., an AI scribe just deployed in a hospital has, by definition, limited pre-change data).

To this end, this work introduces *asymptotic e-detectors* to adaptively monitor a change in a single, potentially unstructured, data stream. Asymptotic e-detectors play an approximation of the **Changepoint-Detection-Game**, where the pre-change distribution is approximated by a finite calibration dataset. Following the foundational work of Chu et al. [14], we show that this game controls the ARL_0 asymptotically, as the size of the calibration dataset $m \rightarrow \infty$; empirically, we find that ARL_0 is well-controlled even for calibration datasets as small as $m = 30$. Having set up the game, we then introduce various betting strategies: bettors that detect black-box shifts using embeddings, bettors that detect interpretable shifts by iteratively searching over candidate concepts using LLMs, and bettor ensembles. Across three real-world text/image data streams, we find that adaptive monitoring procedures outperform their fixed counterparts. When the true shift is sparse and interpretable, using LLMs to learn which concepts to monitor can not only recover the shifted concepts but also substantially accelerate detection speed. Code for reproducing the results in this work is available at <https://github.com/jjfenglab/monitoring-by-betting>.

2 RELATED WORK

Monitoring LLMs and GenAI in practice. The most widespread practice for monitoring drifts in LLM applications involves measuring performance of the system outputs in response to system inputs, often approximated using LLM-as-a-judge [17, 15, 35, 66]. However,

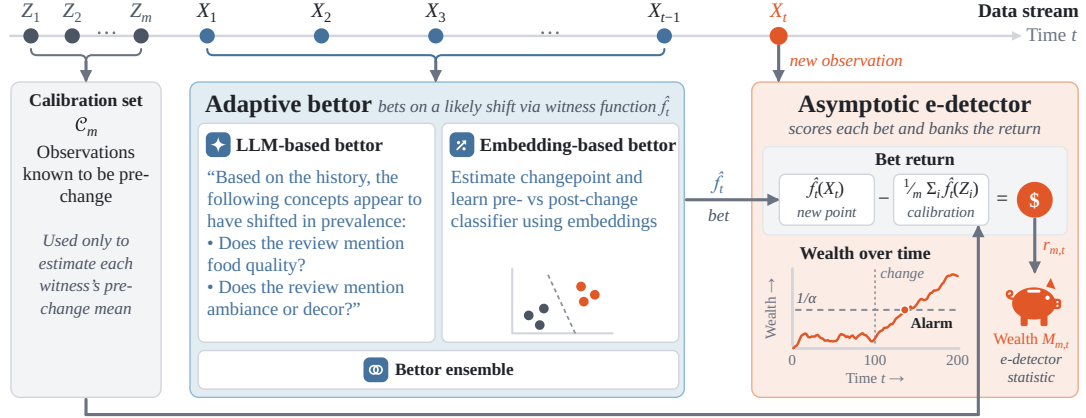


Figure 1: Adaptive monitoring procedure overview: At each time step, an adaptive better uses past observations to bet on how the data may have shifted. When the next observation arrives, the bet’s return is computed by comparing it with a calibration set, and an alarm is raised once the accumulated wealth crosses a threshold.

these approaches require the metrics to be pre-specified and thus suffer from the sensitivity–breadth trade-off, such as failing to include or poor alignment with actual drift directions [72, 60, 20]. The methods developed in this paper can thus provide additional flexibility for monitoring performance or detecting more general shifts. Some works have used e-values to test for the correctness of LLM outputs [18, 58, 73], but not for monitoring unstructured data for changepoints. We note that other specialized testing approaches designed solely to test shifts in LLM behavior—such as repeatedly using the same test set [13] or actively constructing prompts for testing [75, 10, 11]—are outside the scope of this work.

Changepoint detection procedures. There is a vast literature on CUSUM or Shiryaev–Roberts (SR) procedures for changepoint detection, from classical (generalized) likelihood ratio procedures [64] to more recent classifier two-sample test (C2ST)-based ones [49, 27]. However, these classic methods often simulate or analytically derive the false alarm-controlling threshold by assuming a simple, parametric structure on the pre-change distribution or requiring the pre-change distribution to be known, which is not practical for monitoring unstructured inputs and outputs of GenAI systems. Recently, e-detectors [62], which generalize classic changepoint detection procedures to composite and nonparametric settings, have become increasingly popular with the benefit that thresholding at $1/\alpha$ guarantees that ARL_0 is at least $1/\alpha$. E-detectors can be constructed as CUSUM and SR procedures based on e-processes, which generalize the notion of test (super)martingale. CUSUM and SR variants of conformal testing martingales (CTMs) [67, 48] can thus also be viewed as e-detectors on a filtration of conformal p-values. Compared with standard CTMs which conventionally assume a pre-specified conformity score, e-detectors allow all quantities entering the betting function to be adaptive provided they are

predictable, including post-change estimates, betting sizes, and the witness function [34, 53, 61, 55, among others]. Weighted CTMs [48] allow for monitoring an adaptive null hypothesis, but here we focus on adaptively learning the alternative hypothesis. Nevertheless, existing e-detector methods are restricted to data streams with specific structures that are amenable to martingale-like requirements. To our knowledge, there are no e-detector methods for the most common and basic task encountered by GenAI systems: monitoring a single, unstructured data stream. To address this technical gap, this work introduces an extension of the e-detector framework that relaxes the strict martingale requirements.

Interpretable explanations of shifts. Existing work on interpreting data shifts primarily focuses on providing post-hoc explanations [32, 33, 51], which can build trust in AI monitoring systems [42, 19] and facilitate the diagnosis and correction of the underlying model [8]. However, post-hoc explanations may not be faithful to the shift detected originally using black-box methods [57]. Work on white-box changepoint detection methods is extremely limited [69] and requires the set of potential shift directions to be pre-specified. White-box anomaly/Out-of-Distribution (OOD) detection takes a similar approach, using natural language to pre-specify the likely distribution/definition of an OOD sample [23, 7, 16, 74]. However, pre-specification carries a major risk of missing the actual shift, as future shifts are typically unknown in practice. As such, this work considers learning interpretable directions along which to detect shifts, which draws on earlier works learning natural language explanations to classify between two data distributions [71, 72, 21] and searching across the infinite space of concepts to maximize the performance of a Concept Bottleneck Model [37, 25].

3 THE MONITORING GAME

Suppose we sequentially observe independent random variables $(X_1, X_2, \dots)$ with $X_t \in \mathcal{X}$ for all $t \geq 1$, whose joint distribution is denoted by $\mathbf{P}$. Let $(\mathcal{F}_t)_{t \geq 0}$ be the natural filtration with $\mathcal{F}_t = \sigma(X_1, \dots, X_t)$. Nonparametric sequential changepoint detection can be formally defined as monitoring the null hypothesis of there being no changepoint and the alternative of there being a single changepoint, i.e.,

$$H_0 : \exists P, \text{ s.t. } \mathbf{P} = P^\infty, \quad (1)$$

$$H_1 : \exists \tau^* \geq 1 \text{ and } Q \neq P \text{ s.t. } \mathbf{P} = P^{\tau^*} \times Q^\infty, \quad (2)$$

where P and Q are unknown. For a monitoring procedure with stopping time τ , we characterize its operating characteristics through its ARL_0 and Expected Detection Delay (EDD), which are analogs of Type I and II error in classical hypothesis testing and formally defined as

$$\text{ARL}_0 = \mathbb{E}_{H_0}[\tau], \quad (3)$$

$$\text{EDD} = \mathbb{E}_{H_1}[\tau - \tau^* | \tau > \tau^*], \quad (4)$$

where $\mathbb{E}_{H_0}$ and $\mathbb{E}_{H_1}$ denote expectation under P^∞ and $P^{\tau^*} \times Q^\infty$, respectively. Drawing on the e-detector framework, the key to controlling ARL_0 is to design a proper **Changepoint-Detection-Game** that ensures there is no winning strategy under the null. To this end, we begin with describing e-detectors for playing an idealized form of the game and then how their extension—*asymptotic* e-detectors—can play the approximate game in real-world settings. We defer all proofs to Appendix A.

3.1 E-detectors for the idealized game

E-detectors are a general class of sequential tests for the stated changepoint detection hypotheses in the following sense:

Definition 1 (E-detector). *Let $M = (M_t)_{t \geq 0}$ denote a nonnegative adapted process with respect to a filtration $\mathcal{F} = (\mathcal{F}_t)_{t \geq 0}$. The process M is an e-detector for H_0 if $\mathbb{E}_{H_0}[M_\tau] \leq \mathbb{E}_{H_0}[\tau]$ for all $\tau \in \mathcal{T}$, where $\mathcal{T}$ is the set of all stopping times with respect to $\mathcal{F}$.*

From this definition, it immediately follows that for any stopping time $\tau = \inf\{t \geq 1 : M_t \geq 1/\alpha\}$, the ARL_0 is guaranteed to be no smaller than $1/\alpha$.

Despite this abstract definition, e-detectors constructed from a game-theoretic setup can be defined quite concretely as the aggregate wealth of a bettor playing the **Changepoint-Detection-Game**. In particular, suppose there is a sequence of subgames that are smaller versions of the **Changepoint-Detection-Game** and begin at each timepoint with 1 unit of wealth. For each of

these subgames, we define a corresponding e_j -process that represents the bettor's wealth in the subgame starting from candidate changepoint j , i.e.,

Definition 2 (e_j -process). *For any $j \geq 1$, $E^{(j)} = (E_t^{(j)})_{t \geq 0}$ is an e_j -process for H_0 if it is a nonnegative adapted process satisfying $E_0^{(j)} = \dots = E_{j-1}^{(j)} = 1$, and*

$$\mathbb{E}_{H_0} [E_\tau^{(j)} | \mathcal{F}_{j-1}] \leq 1, \forall \tau \in \mathcal{T}.$$

In words, this definition states that each e_j -process represents a subgame where there is no way that the bettor can expect to accumulate wealth under the null. One can then show that valid e-detectors can be constructed through various aggregations of wealth across subgames: based on the classical Shiryaev–Roberts (SR) procedure [63, 56], the SR e-detector is defined as the total wealth across all subgames. Based on Page [46], the CUSUM e-detector is defined as the maximum wealth across all these subgames.

To obtain valid e-processes, the key question is then how to design this betting game. For each unit of wealth, an aggressive bettor could bet all their wealth in hope of a large return rate r_t , but this strategy runs the risk of bankruptcy. In practice, we require each bettor provide a (positive and predictable) betting fraction λ_t that is small enough to avoid betting *all* their wealth; that is, to bound the return rate above zero, $1 + \lambda_t r_t > 0$, for all t (details in Section 3.2). Next, suppose for simplicity that the rate of return r_t is the same across all active subgames. To ensure there is no winning under the null, we require the expected return rate not to exceed zero under the null, i.e.,

Assumption 1. *The return process $r = (r_t)_{t \geq 1}$ adapted to filtration $\mathcal{F}$ satisfies $\mathbb{E}_{H_0} [r_t | \mathcal{F}_{t-1}] \leq 0$ for all t .*

Consequently, the j -th e-process is the cumulative product $E_t^{(j)} := \prod_{s=j}^t (1 + \lambda_s r_s)$, which is a supermartingale under H_0 . The SR e-detector $(M_t^{\text{SR}})_{t \geq 0} = (\sum_{j=1}^t E_t^{(j)})_{t \geq 0}$ can be calculated with the recursion

$$M_t^{\text{SR}} = (1 + M_{t-1}^{\text{SR}})(1 + \lambda_t r_t) \text{ with } M_0^{\text{SR}} = 0; \quad (5)$$

the CUSUM e-detector can be derived using a similar recursion.

We can now set up the **Changepoint-Detection-Game** as follows. Let $\hat{f}_t : \mathcal{X} \rightarrow \mathbb{R}$ be a $\mathcal{F}_{t-1}$ -measurable witness function selected by the bettor, which can be determined however adaptively using any or all of the past data. In an ideal world where the true pre-change distribution P is known, it is natural to define the game's return as the centered quantity

$$r_t = \hat{f}_t(X_t) - \mathbb{E}_{H_0}[\hat{f}_t(X_t) | \mathcal{F}_{t-1}] \quad (6)$$

to satisfy Assumption 1. Then under this ideal game design, the statistics M^{SR} and M^{CU} are valid e-detectors. However, without access to the true pre-change distribution, we cannot obtain the exact value of $\mathbb{E}_{H_0}[\hat{f}_t(X_t) \mid \mathcal{F}_{t-1}]$ and thus cannot form a valid e-detector. The question is then how we can still design monitoring processes that control the ARL_0 .

3.2 Asymptotic e-detectors for the approximate game

As the idealized **Changepoint-Detection-Game** is not feasible in practice, an alternate approach is needed. To this end, we draw on the asymptotic changepoint detection framework introduced in [14], which introduced the notion of a calibration dataset and a new asymptotic framework defined with respect to the size of the calibration dataset; this framework foundationally shifted the changepoint detection field at the time, as it drastically simplified approaches to obtaining (asymptotic) control of the ARL_0 . Inspired by this work, we introduce a similar asymptotic framework.

Suppose we have access to a calibration set $\mathcal{C}_m = \{Z_1, \dots, Z_m\}$ of size m sampled IID under the pre-change distribution P . Intuitively, we could estimate $\mathbb{E}_{H_0}[\hat{f}_t(X) \mid \mathcal{F}_{t-1}]$ with the empirical mean over $\mathcal{C}_m$ and thus approximate the idealized return r_t with

$$r_{m,t} = \hat{f}_t(X_t) - \frac{1}{m} \sum_{i=1}^m \hat{f}_t(Z_i). \quad (7)$$

Plugging $r_{m,t}$ in place of r_t of an SR e-detector, we define

$$M_{m,t} = (1 + M_{m,t-1})(1 + \lambda_t r_{m,t}) \quad (8)$$

for $t \geq 1$ and with $M_{m,0} = 0$. Intuitively, as $m \rightarrow \infty$, we expect the conditional expectation of $r_{m,t}$ to converge to zero, and thus $M_{m,t}$ to behave like an e-detector, under one critical requirement: the witness function must be fully independent of the calibration dataset, so that the empirical average in (7) is a uniformly consistent estimate of the true expectation under the null.

Motivated by this, we introduce the notion of an *asymptotic e-detector* as follows.

Definition 3 (Asymptotic e-detector). *For each calibration size m , let $\mathbf{M}_m = (M_{m,t})_{t \geq 1}$ be a nonnegative process adapted to $\mathcal{G}_m = (\mathcal{G}_{m,t})_{t \geq 0}$ with $\mathcal{G}_{m,t} = \sigma(\mathcal{F}_t, \mathcal{C}_m)$. Then $\mathbf{M}_m$ is an asymptotic e-detector for H_0 if, for the stopping time $\tau_m = \inf\{t : M_{m,t} \geq 1/\alpha\}$,*

$$\limsup_{m \rightarrow \infty} (\mathbb{E}_{H_0}[M_{m,\tau_m}] - \mathbb{E}_{H_0}[\tau_m]) \leq 0. \quad (9)$$

Let us further suppose the following regularity conditions are satisfied:

Assumption 2. (i) For all $t \geq 1$, witness functions $\hat{f}_t$ are measurable w.r.t. $\mathcal{F}_{t-1}$ and take values in $[-B, B]$. (ii) All witness functions $\{\hat{f}_t\}_t$ lie in a fixed P -Glivenko-Cantelli class $\mathbb{F}$. (iii) The calibration set $\mathcal{C}_m = \{Z_1, \dots, Z_m\} \stackrel{\text{iid}}{\sim} P$ and independent of the monitoring stream $X_1, X_2, \dots$. (iv) The stopping times $\{\tau_m\}_m$ under H_0 are uniformly integrable.

Condition (i) states that the witness function cannot depend on the calibration dataset or any downstream variable of the calibration data (e.g., the game's return r_t) and the boundedness is satisfied by standard classifier classes on fixed representations. Condition (ii) is a standard complexity assumption on the witness function class. A calibration dataset that satisfies Condition (iii) is often easy to collect in practice. Condition (iv) is a standard regularity assumption; if instead the expected stopping time is unbounded, the ARL_0 trivially exceeds the nominal level.

Given the assumptions above, the following theorem formally states that an asymptotic e-detector that reuses the same calibration set of size m over time remains valid asymptotically as $m \rightarrow \infty$.

Theorem 1. *Consider the detector $(M_{m,t})_{t \geq 1}$ defined by (8) with $\lambda_t \leq 1/(2B)$. Define the stopping time $\tau_m = \inf\{t \geq 1 : M_{m,t} \geq 1/\alpha\}$. Under Assumption 2, $(M_{m,t})_{t \geq 1}$ is an asymptotic e-detector. Consequently, for any $\alpha \in (0, 1)$, $\liminf_{m \rightarrow \infty} \mathbb{E}_{H_0}[\tau_m] \geq 1/\alpha$, i.e., the ARL_0 is asymptotically controlled at level α .*

Finally, we note that while Theorem 1 only characterizes the asymptotic ARL_0 and not the convergence rate itself, we intuitively expect $M_{m,t}$ to converge quite quickly to the idealized e-detector M_t , as martingale increments are defined in terms of conditional expectations and expectations generally converge quickly (in contrast to convergence of conditional probability distributions).

4 BETTING STRATEGIES

Having set up the game, the question is what betting strategy one should employ, i.e., how should one select the witness function in Step 1 of **Changepoint-Detection-Game**? To this end, we extend ideas from Classifier-based Two-Sample Testing (C2ST) [44, 61], which tests for differences between two data distributions by training a classifier to distinguish between them. Compared to classical two-sample testing methods that use pre-specified distance measures, this classification framing is particularly advantageous for unstructured data, as pre-specifying distance for such data modalities is difficult but AI/ML models are effective at learning discriminators for this space.

Consequently, we introduce Classifier-based Betting, in which the bettor adaptively selects/learns classifiers to distinguish pre- and post-change data. Each betting step can be broken down into four decisions:

Classifier-based Betting

- Select data representation $\phi_t : \mathcal{X} \mapsto \mathbb{R}^K$.
- Estimate changepoint $\hat{\tau}_t$.
- Fit a classifier $\hat{p}_t$ between the pre- vs post-change groups with features $\phi_t(X_i)$ and pseudo-labels $Y_i = \mathbb{1}\{i \geq \hat{\tau}_t\}$. Define witness $\hat{f}_t = \hat{p}_t \circ \phi_t$.
- Select scaling factor λ_t .

This work focuses on steps (a)-(c); for simplicity, this work fixes λ_t to be constant and we refer readers to other papers that study adaptive selection of λ_t [61]. We present two general betting approaches that fork at the first decision point of step (a): using embedding representations versus interpretable extractions of the data. Natural choices for steps (b) and (c) emerge after this first decision point. Finally, since step (a) is arguably the biggest factor influencing detection speed, ensembling can be used to hedge against the uncertainty of this choice (note that ensembling preserves validity of the (asymptotic) e-detector); we evaluate ensembling later in Section 5.

4.1 Bettor for black-box shifts

We first consider the class of betting strategies that detects shifts in embedding space, which is useful when one has minimal prior knowledge of the likely shift type and/or is interested in detecting *any* shift rather than a restricted set of shifts. As such, let $\phi_{emb} : \mathcal{X} \mapsto \mathbb{R}^p$ denote the embedding model selected in step (a). For steps (b) and (c), one can then draw on the large family of general-purpose changepoint learning procedures [28, 6] and classification procedures. A simple example—which we use in later experiments—is to (b) find the maximizer $\hat{\tau}_t$ of the classical changepoint test statistic for multivariate mean shifts [5, 4], i.e.,

$$\max_{\tau} \frac{(\tau - 1)(t - \tau + 1)}{t} \left\| \overline{\phi_{emb}(X)}_{1:(\tau-1)} - \overline{\phi_{emb}(X)}_{\tau:t} \right\|^2 \quad (10)$$

where $\overline{\phi_{emb}(X)}_{a:b}$ represents the mean extraction over positions a to b and $\|\cdot\|$ the L_2 norm, and then (c) fit a logistic regression model for the learned split.

4.2 Bettor for interpretable shifts

Next consider a bettor who relies on interpretable bets, which is useful when they have domain knowledge but difficulty pre-specifying the exact set of possible shifts upfront. Restricting to interpretable bets has two benefits: (i) constraining the set of shifts to those of practical

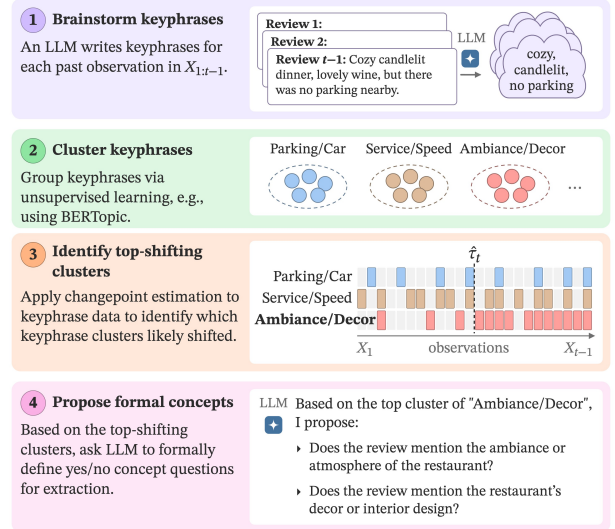


Figure 2: LLM-concept bettor for estimating when and what shifted, using LLM reasoning, unsupervised learning, and statistical changepoint estimation.

interest (e.g., interested in shifts in clinical note quality but not writing style) and thereby reducing false alarms and (ii) injecting prior knowledge to accelerate detection, compared to black-box bets.

Formally, the interpretable bettor must supply a concept-extractor function $\phi_{c_t} : \mathcal{X} \mapsto \mathbb{R}^K$ in step (a), where $\mathcal{C}_t$ is an adaptively selected set of concepts. While concept extraction is generally straightforward (e.g., using an LLM [38, 2]), there is limited work on selecting concepts $\mathcal{C}_t$ to optimize changepoint detection, as detailed in Section 2. Figure 2 outlines an LLM-concept bettor that extends concept selection in [25] to the changepoint setting, in which the AI agent uses statistical tools to identify candidate changepoints and concepts; we found this pipeline to work well in practice and leave further refinement of this pipeline to future work. More specifically, we have an LLM agent first brainstorm potential shifts by generating a “word cloud” of keyphrases describing each observation up to the current time point, cluster keyphrases using BERTopic [29], estimate a working changepoint and top-changing clusters based on keyphrases [22], and have an LLM propose formal yes/no concepts based on the top clusters [25]. More details are in the Appendix.

For steps (b) and (c), one can use any changepoint detection and classification methods but we suggest leveraging the unique advantage of the interpretable bettor: because concept directions are generally disentangled (as opposed to dense embeddings), this bettor can uniquely detect sparse shifts. To this end, we pair interpretable concept bettors with changepoint detection methods for sparse alternatives [22], which jointly learn when and which subset of features shifted (up

to a maximum number of K features) by finding the maximizer $\hat{\tau}_t^{(sparse)}$ of the statistic

$$\max_{\tau, m \in \mathcal{S}_K} \frac{(\tau - 1)(t - \tau + 1)}{t} \left\| \overline{\phi_{c_t}(X)_{m,1:(\tau-1)}} - \overline{\phi_{c_t}(X)_{m,\tau:t}} \right\|^2 - |m| \quad (11)$$

where $\mathcal{S}_K$ represents all possible subsets of $\{1, \dots, p\}$ of sizes 1 to K , and $\overline{\phi_{c_t}(X)_{m,a:b}}$ represents the mean of features restricted to subset m over positions a to b . Likewise, we can use an elastic net penalty when fitting the logistic regression model.

5 EXPERIMENTS

We evaluate the adaptive monitoring-by-betting framework across three experimental settings: a text stream with an injected topical shift (Section 5.1), an image stream from bird photographs with an injected image-attribute shift (Section 5.2), and real-world clinical AI scribe data (Section 5.3). We implement three adaptivebettors that differ in their representation ϕ_t , namely raw embeddings (**Embedding**), sparse autoencoder features of the embeddings (**SAE**), and concepts proposed by **gpt-4.1-mini** (**LLM-concept**), as well as an **Ensemble** that averages the **Embedding** and **LLM-concept** detectors. We report the truncated EDD, the mean number of post-change observations until an alarm or the end of the stream, as a function of the nominal ARL_0 .

For fixed monitors, we run SR on CTMs [67] with fixed (non)conformity scores that trade breadth for sensitivity.¹ **Centroid** scores the distance to the centroid of the calibration set, while **Informed-shift** and **Unrelated-shift** score the cosine similarity with a text-defined direction for the true shift or an unrelated one. As a performance ceiling, **Oracle** uses the difference between the mean post- and pre-change embeddings as its direction.

All experiments are run for 30 replicates for all LLM-based detectors for cost purposes and 100 replicates for all others. All methods use a calibration set of $m = 30$ based on results in Appendix D.1, which verify ARL_0 control across a large range of m values. See Appendix C for full experiment details.

5.1 Monitoring restaurant reviews in CEBaB

We first evaluate how quickly adaptive monitoring procedures can detect a topical shift in text. Using the real-world CEBaB restaurant review dataset [1], we construct disjoint pre- and post-change pools in which reviews mentioning ambiance are strongly upweighted

¹We omit asymptotic e-detectors with a fixed witness, as they are generally more conservative than SR for CTMs with fixed scores, as CTMs provide tight ARL_0 control.

post-change, so that the rate of ambiance mentions increases from 0.52 to 0.88. The **Informed-shift** and **Unrelated-shift** directions use texts about ambiance and about parking (Table C.2).

All procedures with null streams are well calibrated, with simultaneous 95% bands for their empirical ARL_0 that contain or lie above the desired level (Appendix Fig C.1). Compared to fixed monitors, adaptive monitoring speeds up detection (Fig 3a), with **LLM-concept** achieving the lowest EDD. **LLM-concept** quickly learns the correct concept, defined through yes/no questions like “Does the review mention ambiance or atmosphere of the restaurant?” (Fig 3e); this illustrates how it increases sensitivity to the true shift and explains why **LLM-concept** is only slightly slower than the **Oracle** detector. Meanwhile, the adaptive **Ensemble** achieves the second fastest EDD, by hedging across both the interpretable and black-box shifts. Finally, the slow speed of **Informed-shift** is perhaps surprising since it already knows the true concept shift. However, it shows that drafting text to represent candidate drift directions in embedding space can be quite difficult in practice.

5.2 Monitoring bird images from CUB

Next we evaluate the proposed method on detecting shifts in images. Using bird photographs from CUB-200-2011 [68], we construct a set of streams that up-samples birds with yellow bills post-changepoint (prevalence of 0.17 to 0.64) and an analogous set but for white crowns (prevalence of 0.30 to 0.95). We use CLIP embeddings to define the pre-specified directions for the fixed monitors; for instance, the **Informed-shift** direction for streams with a shift in yellow bills uses the exemplar text “a photo of a bird” $\rightarrow$ “a photo of a bird with a yellow bill.”

Across both shift settings, the adaptive **Ensemble** and **LLM-concept** detectors are the first and second fastest, respectively. For the yellow bill shift, the next fastest methods are adaptive **Embedding** and **SAE** monitoring and the **Informed-shift** fixed monitor. For the white crown shift, all the adaptive monitors achieve similar EDD and substantially outperform the fixed monitors; this suggests this shift was easy to learn across all data representations.

The **LLM-concept** bettor learned the correct concepts across most streams (Fig 3f, g). For the yellow bill shift, the LLM learned concepts that exactly coincided, like “Does the image show a bird with yellow beak?”, or overlapping concepts, like “Does the image show a bird with yellow or yellow-orange bill?” Interestingly, for the white crown shift, the LLM learned the correct concept or its opposite, i.e., “Does the image show a bird with a dark head or dark crown?”

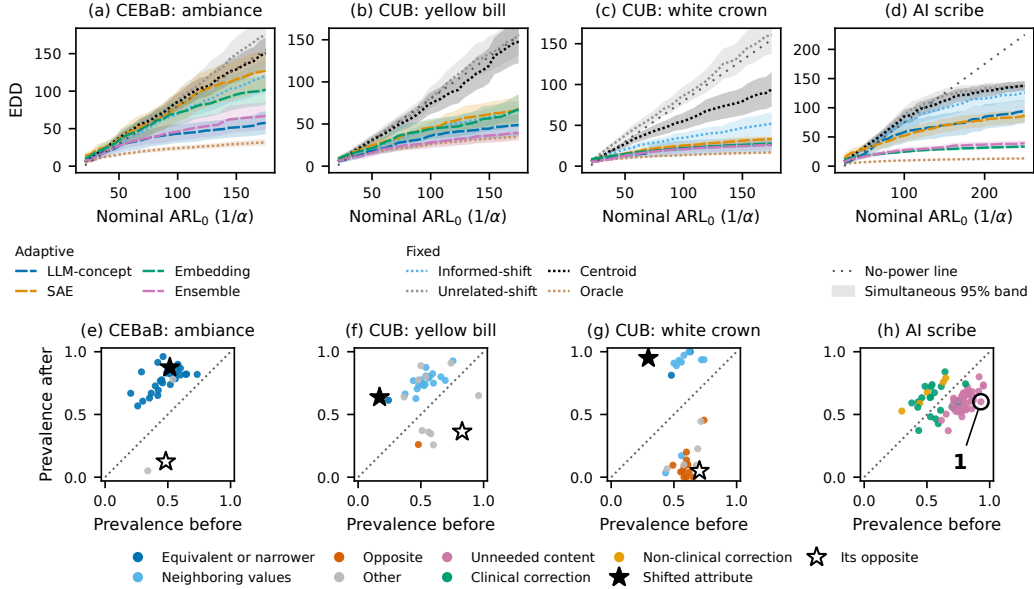


Figure 3: Experimental results. Top: EDD against nominal ARL_0 , where lower is better. Bottom: Concepts learned by the LLM-concept bettor, plotted by prevalence before vs after the changepoint. Concepts in (e-g) are colored by their agreement with the true shifted attribute (stars). Concepts in (h) are colored by themes (Table C.3), as true shifts are unknown for real-world AI deployments. The circled concept in (h) asks whether the draft includes unneeded non-clinical information, such as a review of systems or introductory sentences.

5.3 Monitoring AI scribe data

Working with the AI operations team at UCSF, we monitored shifts in real-world drafts of clinical notes generated by AI scribes and corresponding edits by the healthcare provider. Such shifts can result from changes in the scribe (e.g., model version, prompt) or clinician behavior (e.g., automation bias). Monitoring is needed, as product updates are often unannounced and provide limited insight into how drafts and editing behaviors change. Here we consider a known model update within the Rheumatology department. Streams were constructed by drawing on notes 60 days before and after the update.

Again, we find that all adaptive monitoring procedures were substantially faster than the fixed monitors (Fig 3d). The ranking between the adaptive monitors differs here: **Ensemble** and **Embedding** bettors are now the top-ranking, outperforming **SAE** and **LLM-concept** alone. This change is not entirely unexpected, based on the concepts learned by the LLM-concept bettor (Table C.4): the model update changed the overall structure of the AI draft and made it more concise, so edits that “restructure the note” and “remove unnecessary, redundant, or irrelevant content” became less frequent. Such major changes are more quickly detectable using embeddings, whereas LLM-concept is faster at detecting sparse shifts. Explanations such as these show the utility of the LLM-concept bettor. Indeed, the AI operations team found the interpretable bettor more actionable, as it points to potential directions for future improvements and unanticipated

effects of model updates. These findings also aligned with rheumatologist feedback, as many reported the update improved note structure, reduced unnecessary information, and increased precision of medication information.

6 DISCUSSION

One of the most common and basic tasks when deploying a GenAI system is monitoring unstructured data streams. Whereas existing approaches have relied on detecting shifts in pre-specified directions, this work studies how to adaptively learn what to monitor, given the vast breadth of potential shifts for unstructured data streams. Drawing on testing-by-betting, this work introduces asymptotic e-detectors, which adaptively bet the shift direction at each time step while providing asymptotic false alarm control. Across real-world data streams, adaptive bettors are found to detect shifts faster than fixed monitors: LLM-concept bettors were fastest for sparse interpretable shifts, black-box bettors for broad shifts, and their ensemble achieved (near-)top performance across all settings. Furthermore, by incorporating LLM reasoning, LLM-concept bettors can identify shifts of practical relevance, which often accurately coincided with the true shift in experiments. Finally, while this work shows how reasoning can be incorporated into automated monitoring procedures, it is important to acknowledge that LLMs may identify shifts that are correlated with but are not the actual root cause. Thus expert interpretation of monitoring results is ultimately needed for fully comprehensive

monitoring and diagnosis.

AI use statement

In this work, we used generative AI tools to help with: cleaning and reformatting datasets, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, designing or providing feedback on research methodology or experiments, and implementing methods. We have not used generative AI tools to help with: generating synthetic data sets; helping develop theoretical models or conceptual frameworks; formulating mathematical claims; proposing or refining hypotheses; assisting with translation; supporting qualitative and thematic data analysis; and interpreting results. Additionally, we used generative AI tools to help with: creating figures for illustrating the proposed methods; editing software code; and summarizing existing literature. All AI-assisted work is carefully reviewed and edited by the authors. All LLM-generated code was verified and tested for correctness by authors. All LLM-assisted proofs were checked line by line by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Acknowledgements

The authors thank Nicholas Petrick, Berkman Sahiner, and Gene Pennello (U.S. Food and Drug Administration, Center for Devices and Radiological Health) for helpful discussions and serving as stakeholders for this project.

Disclaimers

The contents represent the views of the author(s) and do not represent the views of, and are not an endorsement by, the U.S. Food and Drug Administration (FDA)/Department of Health and Human Services (HHS) or the U.S. Government. The views, findings, and interpretations contained in this document do not constitute FDA guidance, position on this matter, or legally enforceable requirements.

References

- [1] Eldar David Abraham, Karel D'Oosterlinck, Amir Feder, Y Gat, Atticus Geiger, Christopher Potts, Roi Reichart, and Zhengxuan Wu. CEBaB: Estimating the causal effects of real-world concepts on NLP model behavior. *Adv. Neural Inf. Process. Syst.*, abs/2205.14140, May 2022.
- [2] Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. Large language models are few-shot clinical information extractors. *Empirical Methods in Natural Language Processing*, pages 1998–2022, December 2022.
- [3] Crispin Almodovar, Fariza Sabrina, Sarvnaz Karimi, and Salahuddin Azad. LogFiT: Log anomaly detection using fine-tuned language models. *IEEE Trans. Netw. Serv. Manag.*, 21(2):1715–1723, April 2024.
- [4] Jushan Bai. Common breaks in means and variances for panel data. *J. Econom.*, 157(1):78–92, July 2010.
- [5] Michèle Basseville and Igor V Nikiforov. *Detection of abrupt changes: theory and application*. Prentice-Hall, Inc., USA, February 1993.
- [6] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *J. Math. Imaging Vis.*, 51(1):22–45, January 2015.
- [7] Chentao Cao, Zhun Zhong, Zhanke Zhou, Yang Liu, Tongliang Liu, and Bo Han. Envisioning outlier exposure by large language models for out-of-distribution detection. *Proc. Int. Conf. Mach. Learn.*, 235:5629–5659, June 2024.
- [8] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for HealthCare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, August 2015. ACM.
- [9] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models. *International Conference on*, September 2018.
- [10] Timothée Chauvin, Erwan Le Merrer, François Taïani, and Gilles Tredan. Log probability tracking of llm apis. In *International Conference on Learning Representations*, volume 2026, pages 149447–149465, 2026.
- [11] Timothée Chauvin, Clément Lalanne, Erwan Le Merrer, Jean-Michel Loubes, François Taïani, and Gilles Tredan. Token-efficient change detection in LLM APIs. *arXiv [cs.LG]*, February 2026.
- [12] Jiankang Chen, Tong Zhang, Wei-Shi Zheng, and Ruixuan Wang. TagFog: Textual anchor guidance and fake outlier generation for visual out-of-distribution detection. *Proc. Conf. AAAI Artif. Intell.*, 38(2):1100–1109, March 2024.
- [13] Lingjiao Chen, Matei Zaharia, and James Zou. How is ChatGPT’s behavior changing over time? *Issue 6.2, Spring 2024*, 6(2), March 2024.

- [14] Chia-Shang James Chu, Maxwell Stinchcombe, and Halbert White. Monitoring structural change. *Econometrica*, 64(5):1045–1065, 1996.
- [15] Emma Croxford, Yanjun Gao, Elliot First, Nicholas Pellegrino, Miranda Schnier, John Caskey, Madeline Oguss, Graham Wills, Guan-hua Chen, Dmitriy Dligach, Matthew M Churpek, Anoop Mayampurath, Frank Liao, Cherodeep Goswami, Karen K Wong, Brian W Patterson, and Majid Afshar. Evaluating clinical AI summaries with large language models as judges. *NPJ Digit. Med.*, 8(1):640, November 2025.
- [16] Yi Dai, Hao Lang, Kaisheng Zeng, Fei Huang, and Yongbin Li. Exploring large language models for multi-modal out-of-distribution detection. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5292–5305, Stroudsburg, PA, USA, 2023. Association for Computational Linguistics.
- [17] Databricks. Enhancing LLM-as-a-judge with grading notes. <https://www.databricks.com/blog/enhancing-llm-as-a-judge-with-grading-notes>, July 2024. Accessed: 2025-2-14.
- [18] Guneet S Dhillon, Javier González, Teodora Pan-deva, and Alicia Curth. E-scores for (in) correctness assessment of generative model outputs. *International Conference on Artificial Intelligence and Statistics*, 2025.
- [19] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv [stat.ML]*, February 2017.
- [20] Lisa Dunlap, Krishna Mandal, Trevor Darrell, Jacob Steinhardt, and Joseph E Gonzalez. VibeCheck: Discover and quantify qualitative differences in large language models. *arXiv [cs.CL]*, pages 69177–69205, October 2024.
- [21] Lisa Dunlap, Yuhui Zhang, Xiaohan Wang, Ruiqi Zhong, Trevor Darrell, Jacob Steinhardt, Joseph E Gonzalez, and Serena Yeung-Levy. Describing differences in image sets with natural language. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24199–24208. IEEE, 2024.
- [22] Farida Enikeeva and Zaid Harchaoui. High-dimensional change-point detection under sparse alternatives. *Ann. Stat.*, 47(4):2051–2079, August 2019.
- [23] Sepideh Esmailpour, Bing Liu, Eric Robertson, and Lei Shu. Zero-shot out-of-distribution detection based on the pre-trained model CLIP. *Proc. Conf. AAAI Artif. Intell.*, 36(6):6568–6576, June 2022.
- [24] Louis Faust, Patrick Wilson, Shusaku Asai, Sunyang Fu, Hongfang Liu, Xiaoyang Ruan, and Curt Storlie. Considerations for quality control monitoring of machine learning models in clinical practice. *JMIR Med. Inform.*, 12(1):e50437, June 2024.
- [25] Jean Feng, Avni Kothari, Lucas Zier, Chandan Singh, and Yan Shuo Tan. Bayesian concept bottleneck models with LLM priors. In D Belgrave, C Zhang, H Lin, R Pascanu, P Koniusz, M Ghassemi, and N Chen, editors, *Advances in Neural Information Processing Systems 38*, volume 38, Main Conference, pages 104188–104219, San Diego, California, USA, 2025. Neural Information Processing Systems Foundation, Inc. (NeurIPS).
- [26] Jean Feng, Fan Xia, Karandeep Singh, and Romain Pirracchio. Not all clinical AI monitoring systems are created equal: Review and recommendations. *NEJM AI*, 2(2), January 2025.
- [27] Tingnan Gong, Junghwan Lee, Xiuyuan Cheng, and Yao Xie. Neural network-based CUSUM for online change-point detection. *arXiv [cs.LG]*, October 2022.
- [28] A Gretton, K M Borgwardt, M J Rasch, B Schölkopf, and A Smola. A kernel two-sample test. *The journal of machine learning research*, 13: 723–773, 2012.
- [29] Maarten Grootendorst. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv [cs.CL]*, March 2022.
- [30] Peter Grünwald, Rianne de Heide, and Wouter Koolen. Safe testing. *J. R. Stat. Soc. Series B Stat. Methodol.*, 86(5):1091–1128, November 2024.
- [31] Zaïd Harchaoui, Eric Moulines, and Francis Bach. Kernel change-point analysis. In D Koller, D Schuurmans, Y Bengio, and L Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009.
- [32] Fabian Hinder, Valerie Vaquet, Johannes Brinkrolf, and Barbara Hammer. Model based explanations of concept drift. *arXiv [cs.LG]*, March 2023.
- [33] Fabian Hinder, Valerie Vaquet, and Barbara Hammer. One or two things we know about concept drift-a survey on monitoring in evolving environments. part B: locating and explaining concept drift. *Front. Artif. Intell.*, 7:1330258, July 2024.
- [34] Steven R Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform chernoff bounds via nonnegative supermartingales. *Probab. Surv.*, 17(none):257–317, January 2020.
- [35] Hamel Husain and Shreya Shankar. LLM evals: Everything you need to know – hamel’s blog - hamel husain. <https://hamel.dev/blog/posts/evals-faq/>, January 2026. Accessed: 2026-1-17.

- [36] Olav Kallenberg. *Foundations of modern probability*. Probability Theory and Stochastic Modelling. Springer Nature, Cham, Switzerland, 3 edition, February 2021.
- [37] Pang Wei Koh, Thao Nguyen, Y S Tang, Stephen Mussmann, E Pierson, Been Kim, and Percy Liang. Concept bottleneck models. *ICML*, abs/2007.04612, July 2020.
- [38] Takeshi Kojima, S Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Neural Inf Process Syst*, abs/2205.11916, May 2022.
- [39] Michael R Kosorok. *Introduction to empirical processes and semiparametric inference*. Springer Series in Statistics. Springer, New York, NY, 2008 edition, December 2008.
- [40] Jonas M Kübler, Wittawat Jitkrittum, Bernhard Schölkopf, and Krikamol Muandet. A witness two-sample test. *International Conference on Artificial Intelligence and Statistics*, 2022.
- [41] Jonas M Kübler, Vincent Stimper, Simon Buchholz, Krikamol Muandet, and Bernhard Schölkopf. AutoML two-sample test. *Adv. Neural Inf. Process. Syst.*, pages 15929–15941, June 2022.
- [42] Zachary C Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, September 2018.
- [43] Bo Liu, Li-Ming Zhan, Zexin Lu, Yujie Feng, Lei Xue, and Xiao-Ming Wu. How good are LLMs at out-of-distribution detection? In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8211–8222, 2024.
- [44] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *International Conference on Learning Representations*, 2017.
- [45] Douglas C Montgomery. *Statistical quality control*. John Wiley & Sons, Nashville, TN, 7 edition, 2013.
- [46] E S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100, June 1954.
- [47] Aleksandr Podkopaev, Patrick Blöbaum, Shiva Prasad Kasiviswanathan, and Aaditya Ramdas. Sequential kernelized independence testing. *International Conference on Machine Learning*, 202:27957–27993, 2023.
- [48] Drew Prinster, Xing Han, Anqi Liu, and Suchi Saria. WATCH: Weighted adaptive testing for changepoint hypotheses via weighted-conformal martingales. *arXiv [cs.LG]*, May 2025.
- [49] Nikita Puchkin and Valeriia Shcherbakova. A contrastive approach to online change point detection. *AISTATS*, 206:5686–5713, 2023.
- [50] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. In H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1396–1408. Curran Associates, Inc., 2019.
- [51] Ella Rabinovich, Matan Vetzler, Samuel Ackerman, and Ateret Anaby Tavor. Reliable and interpretable drift detection in streams of short texts. In Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 438–446, Stroudsburg, PA, USA, 2023. Association for Computational Linguistics.
- [52] Aaditya Ramdas and Ruodu Wang. Hypothesis testing with e-values. *arXiv [math.ST]*, 2025.
- [53] Aaditya Ramdas, Johannes Ruf, Martin Larsson, and Wouter Koolen. Admissible anytime-valid sequential inference must rely on nonnegative martingales. *arXiv [math.ST]*, September 2020.
- [54] Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023.
- [55] Leo Richter, Xuanli He, Pasquale Minervini, and Matt J Kusner. An auditing test to detect behavioral shift in language models. *international conference on learning representations*, 2025.
- [56] S W Roberts. A comparison of some control chart procedures. *Technometrics*, 8(3):411–430, August 1966.
- [57] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, May 2019.
- [58] Shuvom Sadhuka, Drew Prinster, Clara Fannjiang, Gabriele Scalia, Bonnie Berger, Aviv Regev, and Hanchen Wang. E-valuator: Reliable agent verifiers with sequential hypothesis testing. *arXiv preprint arXiv:2512.03109*, 2025.
- [59] Glenn Shafer. Testing by betting: A strategy for statistical and scientific communication. *J. R. Stat. Soc. Ser. A Stat. Soc.*, 184(2):407–431, April 2021.
- [60] Shreya Shankar, J D Zamfirescu-Pereira, Bjoern Hartmann, Aaditya Parameswaran, and Ian Arawjo.

- Who validates the validators? aligning LLM-assisted evaluation of LLM outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, volume 4, pages 1–14, New York, NY, USA, October 2024. ACM.
- [61] Shubhanshu Shekhar and Aaditya Ramdas. Non-parametric two-sample testing by betting. *IEEE Trans. Inf. Theory*, 70(2):1178–1203, February 2024.
- [62] Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. E-detectors: a nonparametric framework for sequential change detection. *New England Journal of Statistics in Data Science*, 2024.
- [63] A N Shiryaev. On optimum methods in quickest detection problems. *Theory Probab. Appl.*, 8(1): 22–46, January 1963.
- [64] D Siegmund and E S Venkatraman. Using the generalized likelihood ratio statistic for sequential detection of a change-point. *Ann. Stat.*, 23(1): 255–271, February 1995.
- [65] Alexander Tartakovsky, Igor Nikiforov, and Michele Basseville. *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. CRC Press, August 2014.
- [66] Patrick Vossler, Fan Xia, Yifan Mai, Adarsh Subbaswamy, and Jean Feng. LLMs judging LLMs: A simplex perspective. *International Conference on Artificial Intelligence and Statistics*, 2026.
- [67] Vladimir Vovk, Ivan Petej, Ilia Nouretdinov, Ernst Ahlberg, Lars Carlsson, and Alex Gammernan. Retrain or not retrain: Conformal test martingales for change-point detection. *Symposium on Conformal and Probabilistic Prediction and Applications*, February 2021.
- [68] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-UCSD birds-200-2011 dataset. <https://authors.library.caltech.edu/records/cvm3y-5hh21>, 2011. Accessed: –.
- [69] Maleakhi A Wijaya, Dmitry Kazhdan, Boty Dimanov, and Mateja Jamnik. Failing conceptually: Concept-based explanations of dataset shift. *arXiv [cs.LG]*, April 2021.
- [70] Ruiyao Xu and Kaize Ding. Large language models for anomaly and out-of-distribution detection: A survey. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5992–6012, Stroudsburg, PA, USA, 2025. Association for Computational Linguistics.
- [71] Ruiqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. Describing differences between text distributions with natural language. In *International Conference on Machine Learning*, pages 27099–27116. PMLR, 2022.
- [72] Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. Goal driven discovery of distributional differences via language descriptions. *Advances in Neural Information Processing Systems*, 36:40204–40237, 2023.
- [73] Siyu Zhou, Patrick Vossler, Venkatesh Sivaraman, Yifan Mai, and Jean Feng. Adaptive auditing of AI systems with anytime-valid guarantees. *Advances in Neural Information Processing Systems*, 2026. URL <https://arxiv.org/abs/2605.07002>.
- [74] Jiaqi Zhu, Shaofeng Cai, Fang Deng, Beng Chin Ooi, and Junran Wu. Do LLMs understand visual anomalies? uncovering LLM’s capabilities in zero-shot anomaly detection. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 48–57, New York, NY, USA, October 2024. ACM.
- [75] Xiaoyuan Zhu, Yaowen Ye, Tianyi Qiu, Hanlin Zhu, Sijun Tan, Ajraf Mannan, Jonathan Michala, Raluca Ada Popa, and Willie Neiswanger. Auditing black-box LLM APIs with a rank-based uniformity test. *arXiv [cs.CR]*, June 2025.

CHECKLIST

1. For all models and algorithms presented, check if you include:
 - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes, see Section 3 and 4]
 - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, see Section 3]
 - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes, <https://github.com/jjfenglab/monitoring-by-betting>]
2. For any theoretical claim, check if you include:
 - (a) Statements of the full set of assumptions of all theoretical results. [Yes]
 - (b) Complete proofs of all theoretical results. [Yes, see Appendix A]
 - (c) Clear explanations of any assumptions. [Yes, see Section 3]
3. For all figures and tables that present empirical results, check if you include:
 - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes, <https://github.com/jjfenglab/monitoring-by-betting>]
 - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes, see Appendix B]
 - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes, see Appendix C.2 and figures]
 - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes, see Appendix C.2]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
 - (a) Citations of the creator If your work uses existing assets. [Yes, see References]
 - (b) The license information of the assets, if applicable. [Yes, see Appendix C]
 - (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
 - (d) Information about consent from data providers/curators. [Not Applicable]
 - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes, see IRB info in Appendix C]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
 - (a) The full text of instructions given to participants and screenshots. [Not Applicable]
 - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes, see IRB info in Appendix C]
 - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

A THE ASYMPTOTIC E-DETECTOR FRAMEWORK AND PROOFS

Suppose we observe a stream $X_1, X_2, \dots \in \mathcal{X}$ arriving sequentially and we have a calibration dataset of size m , denoted by $\mathcal{C}_m = \{Z_1, \dots, Z_m\}$, drawn i.i.d. from the pre-change distribution P . Let $\hat{P}_m$ denote the empirical distribution over $\mathcal{C}_m$. The monitoring problem is to test:

$$\begin{aligned} H_0 : & \exists P, \text{ s.t. } \mathbf{P} = P^\infty \\ H_1 : & \exists \tau^* \geq 1 \text{ and } Q \neq P \text{ s.t. } \mathbf{P} = P^{\tau^*} \times Q^\infty. \end{aligned}$$

The operating characteristics of a monitoring procedure are its average run length under the null $\text{ARL}_0 = \mathbb{E}_{H_0}[\tau]$ and its expected detection delay $\text{EDD} = \mathbb{E}_{H_1}[\tau - \tau^* \mid \tau > \tau^*]$, where τ is the stopping time. In this section, we use C to denote constants for shorthand.

A.1 Testing and monitoring by betting

We begin by reviewing e-processes and e-detectors from the testing-by-betting framework. More details can be found in [62] and [52].

Baseline E-process Start with \$1 of initial wealth and, at each time t , bet against H_0 so that the wealth E_t at t is updated by $E_t = E_{t-1} \cdot (1 + \lambda_t r_t)$ where $\lambda_t \in (0, 1)$ is a bet scaling factor and r_t is the per-step return. The bet is *fair* under the null if $\mathbb{E}[r_t \mid \mathcal{F}_{t-1}] \leq 0$. Along with the assumption that $1 + \lambda_t r_t \geq 0$ for any t , $(E_t)_{t \geq 0}$ is a nonnegative supermartingale and thus an e-process under H_0 . By Ville's inequality, $\mathbb{P}_{H_0}(\sup_t E_t \geq 1/\alpha) \leq \alpha$ and thus the false alarm rate is controlled at level α uniformly over time. Under H_1 , the conditional mean of the return r_t is positive and the wealth E_t grows exponentially to trigger an alarm.

E-detector Because τ^* is unknown, no single e-process suffices: one started before the change expends its capital on pre-change observations. Following Shin et al. [62], at each timepoint $j \geq 1$ a fresh \$1 is invested in a new subgame betting against the null starting at index j . The detector is the sum of all subgames:

$$M_t = \sum_{j \leq t} E_t^{(j)}, \quad E_t^{(j)} = \prod_{s=j}^t (1 + \lambda_s r_s^{(j)}) \quad (12)$$

with $M_0 = 0$. When the subgames are e-processes starting at timepoints j , total wealth of all subgames under H_0 cannot outrun the money invested: $\mathbb{E}_{H_0}[M_\tau] \leq \mathbb{E}_{H_0}[\tau]$ for any stopping time τ . Defining the stopping time as $\tau = \inf\{t : M_t \geq 1/\alpha\}$ yields $\text{ARL}_0 = \mathbb{E}_{H_0}[\tau] \geq 1/\alpha$. When $r_t^{(j)}$ does not depend on the subgame index j , the sum reduces to the Shiryaev–Roberts recursion

$$M_t = (1 + M_{t-1}) \cdot (1 + \lambda_t r_t), \quad (13)$$

allowing the total wealth M_t to be updated in a single multiplication per step.

Asymptotic E-detectors As defined in the main manuscript, a process $\mathbf{M}_m = \{M_{m,t}\}_{t \geq 0}$ with an associated stopping time $\tau_m = \inf\{t : M_{m,t} \geq 1/\alpha\}$ is an asymptotic e-detector if $\mathbf{M}_m$ satisfies

$$\limsup_{m \rightarrow \infty} (\mathbb{E}_{H_0}[M_{m,\tau_m}] - \mathbb{E}_{H_0}[\tau_m]) \leq 0. \quad (14)$$

A.2 Proof of Theorem 1

Recall the SR detector which reuses the same calibration set $\mathcal{C}_m$ of size m over time defined by

$$\begin{aligned} M_{m,t} &= \sum_{j=1}^t E_{m,t}^{(j)}, \quad M_{m,0} = 0, \\ E_{m,t}^{(j)} &= \begin{cases} \prod_{s=j}^t (1 + \lambda_s r_{m,s}) & \text{for } t \geq j \\ 1 & \text{otherwise} \end{cases} \\ r_{m,t} &= \hat{f}_t(X_t) - \hat{P}_m \hat{f}_t. \end{aligned}$$

In this subsection, we present the proof of Theorem 1 in the main manuscript that the process $\mathbf{M}_m = \{M_{m,t}\}_{t \geq 0}$ is an asymptotic e-detector with asymptotic ARL_0 control.

Suppose the following standard regularity conditions are satisfied.

Assumption 3 (Boundedness). *The witness functions satisfy $\hat{f}_t(x) \in [0, 1]$ for all $t \geq 1$ and all $x \in \mathcal{X}$ and $\hat{f}_t$ is $\mathcal{F}_{t-1}$ -measurable.*

For simplicity, we assume $\hat{f}_t(x) \in [0, 1]$ which is equivalent to the assumption in the main manuscript up to an affine transformation.

Assumption 4. *The calibration dataset $\mathcal{C}_m$ is a random sample from the pre-change distribution P and is independent of the monitoring stream $X_1, X_2, \dots$*

Assumption 5 (Uniformly integrable stopping time). *The class of stopping time $\{\tau_m\}_{m \in \mathbb{N}}$ is uniformly integrable under H_0 (1). More specifically, the following are satisfied by $\{\tau_m\}_m$:*

1. $\exists$ a finite T^* such that $\forall m \geq 1, \mathbb{E}_{H_0}[\tau_m] \leq T^*$.
2. $\forall \epsilon, \exists \delta$, such that $\forall$ measurable A with $P(A) \leq \delta$ and $\forall m \geq 1, \mathbb{E}_{H_0}[|\tau_m| I_A] \leq \epsilon$.

Note that we study behavior of the procedure under this uniform integrability assumption because if $\mathbb{E}_{H_0}[\tau_m] \rightarrow \infty$, then ARL_0 trivially satisfies the run length requirement.

Assumption 6 ([39]). *All witness functions $\{\hat{f}_t\}$ lie in a fixed P -Glivenko-Cantelli (GC) class $\mathbb{F}$.*

Proof. We decompose the one-step increment of $M_{m,t}$:

$$M_{m,t} - M_{m,t-1} = 1 + D_{m,t} + \xi_{m,t}, \quad (15)$$

where $\xi_{m,t}$ is a martingale difference sequence:

$$\begin{aligned} \xi_{m,t} &= (M_{m,t} - M_{m,t-1}) - \mathbb{E}_{H_0}[M_{m,t} - M_{m,t-1} \mid \mathcal{F}_{t-1}, \mathcal{C}_m] \\ &= \lambda_t (r_{m,t} - \delta_{m,t})(1 + M_{m,t-1}), \end{aligned} \quad (16)$$

where $\delta_{m,t} = \mathbb{E}_{H_0}[r_{m,t} \mid \mathcal{F}_{t-1}, \mathcal{C}_m] = (P - \hat{P}_m) \hat{f}_t$ and $D_{m,t}$ is the drift:

$$D_{m,t} = \lambda_t \delta_{m,t} (1 + M_{m,t-1}). \quad (17)$$

Summing to τ_m and taking expectations:

$$\mathbb{E}_{H_0}[M_{m,\tau_m}] - \mathbb{E}_{H_0}[\tau_m] = \mathbb{E}_{H_0} \left[\sum_{t=1}^{\tau_m} D_{m,t} \right] + \mathbb{E}_{H_0} \left[\sum_{t=1}^{\tau_m} \xi_{m,t} \right]. \quad (18)$$

It is sufficient to show that the right hand side is upper bounded by a term converging to 0.

Martingale term: The partial sum process $S_T := \sum_{t=1}^T \xi_{m,t}$ is a martingale with respect to $\sigma(\mathcal{F}_T, \mathcal{C}_m)$. For $t \leq \tau_m$, $M_{m,t-1} < 1/\alpha$ and, by Assumption 3, $|r_{m,t} - \delta_{m,t}| \leq 2$, so increments are uniformly bounded:

$|\xi_{m,t}| \leq 2(1+1/\alpha)$. By Assumption 5, there exists T^* such that $\mathbb{E}_{H_0}[\tau_m] \leq T^* < \infty$. Then, the Optional Stopping Theorem implies

$$\mathbb{E}_{H_0} \left[\sum_{t=1}^{\tau_m} \xi_{m,t} \right] = 0.$$

Drift term: Since $M_{m,t-1} < 1/\alpha$ for all $t \leq \tau_m$ and $\lambda_t \in (0, 1)$, we have

$$\left| \sum_{t=1}^{\tau_m} D_{m,t} \right| \leq C \sum_{t=1}^{\tau_m} |\delta_{m,t}| \leq C \tau_m \sup_t |(P - \hat{P}_m) \hat{f}_t| := \Delta_m$$

for some positive constant C . Since $\hat{f}_t$ is bounded, $\sup_t |(P - \hat{P}_m) \hat{f}_t|$ is positive and bounded. Together with uniform integrability of τ_m , Δ_m is also uniformly integrable. Moreover, $\Delta_m \rightarrow 0$ in probability since all witness functions lie in a Glivenko-Cantelli class and τ_m is uniformly integrable. Therefore, by Vitali convergence theorem [36],

$$\mathbb{E}_{H_0} \left| \sum_{t=1}^{\tau_m} D_{m,t} \right| \leq \mathbb{E}_{H_0}[\Delta_m] \rightarrow 0,$$

establishing the claim. $\square$

Corollary 1 (Asymptotic ARL_0 control). *Asymptotic e-detectors provide asymptotic ARL_0 control under the null:*

$$\liminf_{m \rightarrow \infty} \text{ARL}_{0,m} = \liminf_{m \rightarrow \infty} \mathbb{E}_{H_0}[\tau_m] \geq \frac{1}{\alpha}. \quad (19)$$

Proof. By definition of $\tau_m = \inf\{t : M_{m,t} \geq 1/\alpha\}$, $M_{m,\tau_m} \geq 1/\alpha$ whenever $\tau_m < \infty$. Hence $\frac{1}{\alpha} \leq \mathbb{E}_{H_0}[M_{m,\tau_m}]$. Since M_m is an asymptotic e-detector, for any $\epsilon > 0$ there exists m_0 such that for all $m \geq m_0$:

$$\frac{1}{\alpha} \leq \mathbb{E}_{H_0}[M_{m,\tau_m}] \leq \mathbb{E}_{H_0}[\tau_m] + \epsilon. \quad (20)$$

Rearranging gives $\mathbb{E}_{H_0}[\tau_m] \geq 1/\alpha - \epsilon$ for all $m \geq m_0$. Since $\epsilon > 0$ was arbitrary, taking the limit inferior yields $\liminf_{m \rightarrow \infty} \mathbb{E}_{H_0}[\tau_m] \geq 1/\alpha$. $\square$

B IMPLEMENTATION DETAILS

B.1 Adaptive betting hyperparameters

Bet scaling factor. The bet scaling factor can adapt over time and should balance between aggressive betting (high λ , faster detection but higher variance) and conservative betting (low λ , slower but more stable). While adaptive tuning of λ_t is possible within the e-detector framework and may improve performance, this work uses a fixed value of $\lambda = 0.5$ and leaves further optimization to future work.

Changepoint learner. The changepoint learner estimates when the distribution shifted, which determines the pseudo-labels for classifier training. The mean-shift estimator (Equation 10) is appropriate when shifts are expected to be dense across features, as in embedding spaces where many coordinates may move together. The sparse estimator (Equation 11) is preferable when shifts are expected to affect only a subset of features, as with interpretable concepts where a small number of attributes may change. In our experiments, we use the mean-shift estimator for the black-box bettor (operating on dense embeddings) and the sparse estimator with maximum subset size $B = 4$ for the interpretable bettor (operating on disentangled concepts).

Classifier. The classifier maps features to a score distinguishing pre- from post-change observations. Any classifier producing calibrated probabilities is suitable; we use logistic regression for its simplicity and interpretability, and apply regularization to prevent overfitting when the data representation is high-dimensional.

Calibration dataset size. The calibration dataset size m controls the accuracy of the empirical approximation to the pre-change distribution. Larger m reduces calibration error and tightens ARL_0 control, but comes with practical costs: more pre-change data must be available before monitoring begins, and for interpretable bettors, the calibration set must be annotated with each proposed concept, incurring LLM query costs linear in m . Fortunately, our experiments found ARL_0 control to be robust across a wide range of calibration sizes; see Appendix D.1 for simulations demonstrating that even $m = 30$ can provide effective control.

Burn-in period. An initial burn-in period is needed so that there is sufficient sample size to meaningfully estimate the changepoint and train the classifier. During burn-in, the e-detector does not place bets, so M_t remains at its initial value of 0. The burn-in period should be set such that data during burn-in is nearly guaranteed to be pre-change, to avoid significant delays in detecting shifts.

B.2 The LLM-concept bettor

The LLM-concept bettor uses a four-stage pipeline that interleaves LLMs and unsupervised clustering of text to propose and select concepts at each time point. This four-step procedure is used instead of a zero-shot LLM pipeline, as the latter is often impractical and far from optimal for monitoring long data streams: an LLM is unlikely to be able to fit the entire history into its context window and the LLM is likely to forget observations in the middle of the stream. Instead, this pipeline combines LLM reasoning with statistical/data-analytic tools, which can be viewed as an agentic process. This leads to a pipeline that essentially conducts two rounds of changepoint detection: steps 1-2 are to brainstorm topics and changepoints, while steps 3-4 are to formalize the concepts, annotations, and changepoint estimation. This pipeline is modeled after the agentic pipeline in [25], which was originally designed for classification and is extended here to the task of changepoint detection.

Step 1: Keyphrase extraction. For each past observation, an LLM is asked to extract a set of keyphrases describing its content according to some **keyphrase-extraction-prompt**. (For image streams, we use a VLLM.) The **keyphrase-extraction-prompt** provides the monitoring team a way to communicate their priorities regarding which general categories of concepts and shifts are of practical interest. Figure B.1 shows an example prompt for the CEBaB experiment. Note that keyphrase extraction should be viewed as a brainstorming step, as it is not formal concept extraction nor concept selection. Rather a shift in the types of keyphrases can provide clues as to the types of shifts that are happening in the data.

```

Extract all concepts mentioned this restaurant review. These will be candidates for a central
dictionary of concepts for characterizing restaurant reviews, and will eventually be input
features to a concept bottleneck model for monitoring changes in reviews written over time.
Only list factors explicitly mentioned in the review. Do not infer or add details unless
explicitly stated.
For each concept mentioned in the review, provide:
1. A word or phrase to represent that concept (1-2 words)
2. Synonyms or alternative phrasings
3. Broader generalizations that capture similar concepts
Requirements:
- Each descriptor should be <=2 words. One word descriptors are encouraged.
- The descriptor should state distinguishing features. Focus on factors relevant to the dining
experience.
- List specific terms, synonyms, and increasingly general concepts.
- Separate items with commas

Review(s): "{note}"

Output Format:
{ "reasoning": "...", "keyphrases": [...] }

```

Figure B.1: Keyphrase extraction prompt (abbreviated) for CEBaB experiment.

Step 2: Topic modeling and initial changepoint estimation. Keyphrases are clustered into topics using BERTopic [29]. To speed up BERTopic, one can further use an online variant, as long as each of its internal components allow for online learning. For instance, our experiments use BERTopic with incremental PCA

and clusters keyphrases into topics with mini-batch k -means. Then based on the extracted keyphrases, each observation is mapped to all its cluster assignments (binary vector with 1s for clusters that are present and 0s for clusters that are absent). We then estimate a working changepoint $\hat{\tau}^{(topic)}$ using the sparse changepoint statistic (Equation 11) on this topic representation, which identifies both the candidate changepoint and the topics with the largest prevalence shifts.

Step 3: Concept proposal. An LLM is prompted to propose formal yes/no concept questions based on the top-shifted BERTopic topics and their representative documents. For each of the top topics, the LLM prompt includes each topic’s keywords, representative documents, and how the prevalence of the topic has shifted. Figure B.2 shows one such example for CEBaB.

```
We are monitoring for concepts that shift in prevalence in restaurant reviews that are posted
over time. A concept has to be defined in terms of a yes/no question of the form "Does the
review mention the...?". [...] We have 3 concepts so far:
* X0 = Does the review mention the ambience or atmosphere of the restaurant, including aspects
like crowding, romantic setting, or a pleasant/charming atmosphere?
* X1 = Does the review mention the food being well-prepared or the variety of food offered?
* X2 = Does the review mention the ambience or atmosphere of the restaurant?

To come up with the 4th concept, we conducted a data-driven analysis using topic modeling
(BERTopic) on extracted keyphrases. Below are the topics that show the largest shifts in
prevalence over time (excluding topics already used for existing concepts). [...]

* topic 22 (shift: -0.21): "experience" (n=7), "staff performance" (n=4), "friendly staff"
(n=3), "crowded" (n=3), "overall experience" (n=3), "positive experience" (n=2), "good
experience" (n=2)
* topic 9 (shift: +0.18): "atmosphere" (n=33)
* topic 11 (shift: +0.15): "dining experience" (n=22), "dining environment" (n=2), "dining
occasion" (n=1), "dining time" (n=1), "dining purpose" (n=1), "dining service" (n=1),
"restaurant environment" (n=1)
* topic 0 (shift: +0.15): "price" (n=13), "expensive" (n=8), "cost" (n=6), "overpriced" (n=4),
"high cost" (n=4), "highly rated" (n=2), "pricey" (n=2)

[...]
YOUR TASK: Propose 2 candidate concepts to use as inputs into a final interpretable changepoint
estimation method, based on the topic analysis. [...]

Output Format:
{ "reasoning": "...", "concepts": [{"definition": "...", "topic_indices": [...]}, ...] }
```

Figure B.2: Concept proposal prompt (abbreviated) for CEBaB experiment. Top keyphrase clusters (topics) are shown from an example replicate.

Step 4: Concept annotation and selection. For each of the proposed concepts, the LLM is asked to extract the answer to the formal yes/no concept question for each observation in the history. Applying sparse changepoint estimation (Equation 11) to the LLM-extracted concepts, one finally obtains the final changepoint estimate $\hat{\tau}_t$ for formalized concepts, as well as the selection of top concepts. These selected concepts, coupled with the LLM, then define the data representation function ϕ_{c_t} used for betting. Figure B.3 shows an example annotation prompt.

```
You will be given a restaurant review. Answer the following question with a probability from 0
to 1. Answer with 0 or 1 when possible, but a probability between 0 and 1 is allowed if needed
to express uncertainty.

Question: {prompt_question}

Restaurant review: {text}

Example answer:
{ "reasoning": "...", "answer": 1 }
```

Figure B.3: Concept annotation prompt for the CEBaB experiment.

Finally, we note that the concept extractor must not itself exhibit temporal shifts, as this would violate ARL_0

Table C.1: Summary of experimental settings across the three datasets.

	CEBaB	CUB: yellow bill	CUB: white crown	AI scribe
Data	Restaurant reviews from CEBaB	Bird images from CUB-200-2011	Bird images from CUB-200-2011	HPI section of Rheumatology clinical notes
Shifted attribute	Whether review mentions ambiance	Whether bird has yellow bill in image	Whether bird has white crown in image	Clinical notes from 60 days before vs after model update on 2026-02-25
Prevalence pre→post	0.52→0.88	0.17→0.64	0.30→0.95	N/A
Embedding model	MiniLM	CLIP ViT-B/32	CLIP ViT-B/32	MiniLM
Embedding dim d	384	512	512	384
n_{pre}	50	50	50	100
n_{post}	300	300	300	150
Burn-in	30	30	30	75

guarantees. In our implementation of the LLM concept extractor, we ensure stationarity by processing each observation independently, extracting each concept independently, and retrying failed LLM queries rather than imputing default values (as query failure rates may also vary over time). Future work may consider how to ensure temporal consistency of LLM concept extraction over longer periods of time, where LLM providers may silently switch model versions or hardware.

C EXPERIMENTS

C.1 Experimental settings

Table C.1 summarizes the experimental setup for each dataset. All experiments use 100 simulation replicates for monitors other than **LLM-concept** and **Ensemble** and 30 replicates for **LLM-concept** and **Ensemble** due to cost constraints. On CEBaB and CUB, the non-LLM bettors also run on 100 null streams to illustrate ARL_0 control. The code for the **LLM-concept** bettor is carefully implemented such that control of the ARL_0 for non-LLM bettors also implies ARL_0 control for the **LLM-concept** bettor. In particular, the LLM only affects the bettor through the choice of features, which is made from past observations, so the witness at step t is still fit before X_t is observed and never uses the calibration set, as Conditions (i) and (iii) of Assumption 2 require. Each annotation call labels a single observation for a single concept, so X_t and the calibration observations are annotated by the same procedure, independently of one another. Other than the choice of features, the **LLM-concept** bettor uses the same classifier, plug-in centering on the calibration set, and SR recursion as the **Embedding** and **SAE** bettors, whose simultaneous 95% bands for the empirical ARL_0 on the null streams contain or lie above the nominal ARL_0 (Figures C.1 and C.2).

Pool construction. For CEBaB and CUB, we construct disjoint pre-change (p0) and post-change (p1) pools and inject a controlled shift by upsampling observations with a target attribute in the post-change pool. Specifically, p1 samples are drawn with probability proportional to $\sigma(\beta_0 + \beta \cdot a(x))$, where $a(x) \in \{0, 1\}$ is the binary attribute indicator and σ is the logistic function. We use $\beta = 4$, $\beta_0 = -4$ for CEBaB and $\beta = 6$, $\beta_0 = -4$ for CUB. For CEBaB, we first sample one review variant per original review from the CEBaB counterfactual edits, ensuring each unique review appears at most once across all pools. For CUB, the population is restricted to seabird genera (Gull, Tern, Albatross, Jaeger, Puffin, Cormorant, Fulmar, Kittiwake, Auklet, Merganser, and Pelican), comprising 1,945 images across 34 species. For the white crown attribute, we use consensus labels that require an image’s annotation to agree with the majority label of its species, as raw CUB annotations exhibit within-species variation. In every experiment, a stream draws its calibration set and pre-change observations from the pre-change pool without replacement and its post-change observations from the post-change pool, and null streams draw all observations from the pre-change pool. For CEBaB, the reviews are split by original review into three disjoint sides, with 20% for training the **SAE**, 55% for the pre-change pool and 25% for the post-change pool, and the pre-change pool and the **SAE** side keep each variant with equal probability. For CUB, the images are randomly permuted and the first 730 form the pre-change pool, the post-change pool is 300 images drawn from the remaining

Table C.2: Text exemplars used to define pre-specified drift directions for fixed monitors. Each direction is defined by an in-class (IC) and out-of-class (OC) text description, embedded to form a unit drift vector.

Dataset	Direction	IC text	OC text
CEBaB	Informed	a typical restaurant review describing the customer’s experience of the visit	a restaurant review that comments on the ambiance or decor of the restaurant
	Unrelated	a typical restaurant review describing the customer’s experience of the visit	a restaurant review that mentions parking the car near the restaurant
CUB (yellow bill)	Informed	a photo of a bird	a photo of a bird with a yellow bill
	Unrelated	a photo of a bird	a photo of a bird perched on a wire or fence post
CUB (white crown)	Informed	a photo of a bird	a photo of a bird with a white crown
	Unrelated	a photo of a bird	a photo of a bird perched on a wire or fence post
AI scribe	Informed	an initial draft from an AI scribe with major editing by the clinician, deleting sections and removing verbose and irrelevant content	an initial draft from an AI scribe with major editing by the clinician, with minimal to light editing like medication name fixes
	Unrelated	an initial draft from an AI scribe with major editing by the clinician	an initial draft from an AI scribe with major editing by the clinician where draft becomes less accurate and needs more editing

1,215, and the 915 images in neither pool train the SAE.

AI scribe. The AI scribe dataset consists of History of Present Illness (HPI) sections from rheumatology encounters, collected under IRB approval at UCSF. Each observation is the structured difference between the AI-generated draft and the clinician-edited final note. The shift corresponds to a vendor model update intended to improve performance for rheumatology; the vendor did not disclose specifics of the update beyond its date, though clinicians reported subjective improvements in draft quality. Because this is a real-world temporal shift without ground-truth labels, prevalence statistics are not available. The pre- and post-change pools hold the HPI sections from the 60 days before and the 60 days after the update, 300 and 190 notes, respectively. Each edit is a word-level deletion, insertion or replacement between the draft and the signed note, shown with three words of surrounding context. The SAE is trained on a random 20% of the pre-change notes, which streams never draw.

Adaptive bettor settings. Every adaptive bettor bets $1 + \lambda(\hat{f}_t - \hat{f}_{\text{null}})$ with $\lambda = 0.5$ against a calibration set of $m = 30$ pre-change observations. The changepoint statistics use whitened features and a minimum block length of 10, and the SAE bettor, like the LLM-concept bettor, selects a subset of its latent features with the sparse statistic. The classifier is a logistic regression on standardized features with balanced class weights, fit with a ridge penalty for the Embedding bettor and no penalty otherwise. The SAE is a TopK sparse autoencoder with 32 latent features and $K = 3$. The Ensemble averages the e-detectors M_t of the LLM-concept and Embedding bettors on the same stream, as the average of e-detectors is itself an e-detector. BERTopic for the LLM-concept bettor clusters keyphrases into 25 topics at every update.

Baselines. The fixed monitors run the Shiryaev–Roberts procedure on conformal p-values with the Simple Jumper betting strategy (jump rate 0.01) [67]. Half of the calibration set serves as the reference set for the nonconformity score and the other half seeds the conformal p-values. Centroid scores an observation by its cosine distance to the centroid of the reference set, and the guided variants score it by its cosine similarity with a fixed direction. For Informed-shift and Unrelated-shift, the direction is the normalized difference between the embeddings of the two descriptions in Table C.2, using the text encoder that matches the observation embedding. For Oracle, the direction is the normalized difference between the mean embeddings of the post- and pre-change pools.

C.2 Additional results

ARL₀ is controlled. Figures C.1 and C.2 show the empirical ARL₀ on the CEBaB and CUB null streams.

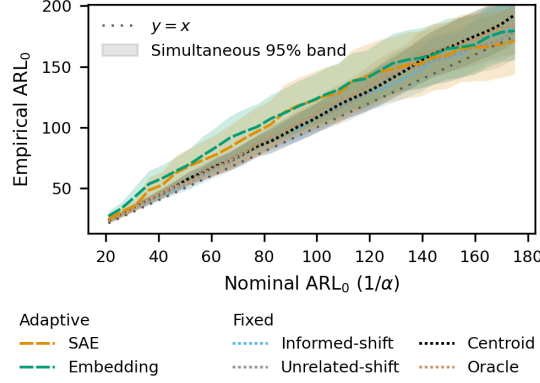


Figure C.1: Empirical ARL_0 against nominal ARL_0 on the 100 CEBaB ambient null streams. Shaded regions are simultaneous 95% bands over streams, and the dotted line marks empirical ARL_0 equal to nominal ARL_0 .

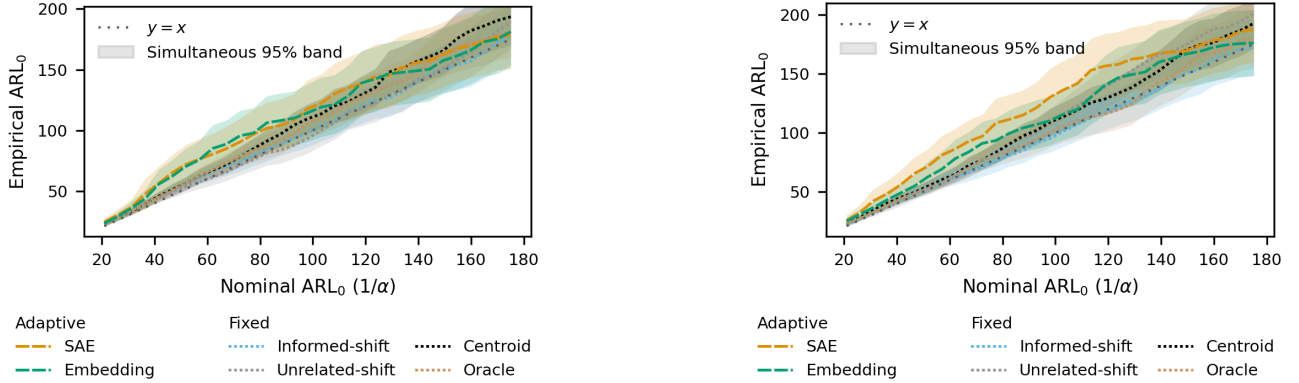


Figure C.2: Empirical ARL_0 against nominal ARL_0 on the CUB yellow bill (left) and white crown (right) null streams, over 100 null streams per setting. Shaded regions are simultaneous 95% bands over streams, and the dotted line marks empirical ARL_0 equal to nominal ARL_0 .

Learned concepts for AI scribe experiment. Table C.3 groups the AI scribe concepts of Figure 3(h) into themes, and Table C.4 lists the concepts of four AI scribe streams.

Simultaneous confidence bands. We show each performance curve with a bootstrapped 95% simultaneous confidence band.

Compute. All experiments ran on CPUs on an internal academic Slurm cluster. Each monitoring stream ran as its own single-node job with 2 CPU cores and 10 GB of memory, and the sentence embeddings and SAE training also ran on CPU. LLM calls went to `gpt-4.1-mini` through an institution-approved Azure OpenAI deployment.

Licenses. CEBaB is distributed under CC BY 4.0, and CUB-200-2011 is restricted to non-commercial research and educational use. The MiniLM sentence encoder, the sentence-transformers library, and HypotheSAEs are released under Apache-2.0, and BERTopic and the CLIP code are released under MIT. We accessed `gpt-4.1-mini` under the Azure OpenAI service terms, and the AI scribe data were used under IRB approval and are not publicly released.

Cost. At `gpt-4.1-mini` prices of \$0.40 and \$1.60 per million input and output tokens, the median LLM-concept stream cost \$2.26 on CEBaB, \$3.42 on CUB yellow bill, \$1.64 on CUB white crown and \$13.59 on AI scribe. A stream took a median of 27 minutes, ranging from 11 minutes to 1.8 hours.

Table C.3: Themes of the AI scribe concepts in Figure 3(h), over the four concepts active at the end of each of 18 streams. An LLM judge (gpt-4.1-mini) assigns each concept to a theme from its text alone, using a fixed codebook based on the vendor’s stated purpose for the update and the clinicians’ reports of editing less. The first two themes come from these reports; the third theme is named after reading concepts that fit neither. Before $\rightarrow$ after shows the mean fraction of notes for which a concept is annotated as present, before and after the changepoint.

Theme	Codebook definition	# Concepts	Before $\rightarrow$ after	Example concepts (before $\rightarrow$ after)
Unneeded content	Draft content that does not belong in the note because it is unnecessary, redundant, verbose, non-clinical, introductory or identifying, or the removal, condensing or restructuring of such content, whether clinical or not	46	0.79 $\rightarrow$ 0.60	Does the initial draft include unnecessary or irrelevant non-clinical information such as review of systems, introductory sentences, or personal details? (0.93 $\rightarrow$ 0.60) Does the edit remove redundant or unnecessary note sections (including review of systems or initial summary)? (0.86 $\rightarrow$ 0.54)
Clinical correction	Clinical content (diagnoses, symptoms, medications, treatment) that the clinician adds, corrects or clarifies, or that the draft is missing	21	0.56 $\rightarrow$ 0.61	Does the edit correct or improve diagnostic information? (0.38 $\rightarrow$ 0.59) Does the initial draft miss or have incorrect medication, treatment, or procedure details? (0.59 $\rightarrow$ 0.43)
Non-clinical correction	Non-clinical information (patient or social history, follow-up, administrative or demographic details) that the clinician adds, corrects or clarifies, or that the draft is missing	5	0.51 $\rightarrow$ 0.67	Does the edit improve the accuracy and completeness of patient, social history, or provider demographic information? (0.50 $\rightarrow$ 0.68)

D ADDITIONAL EXPERIMENTS

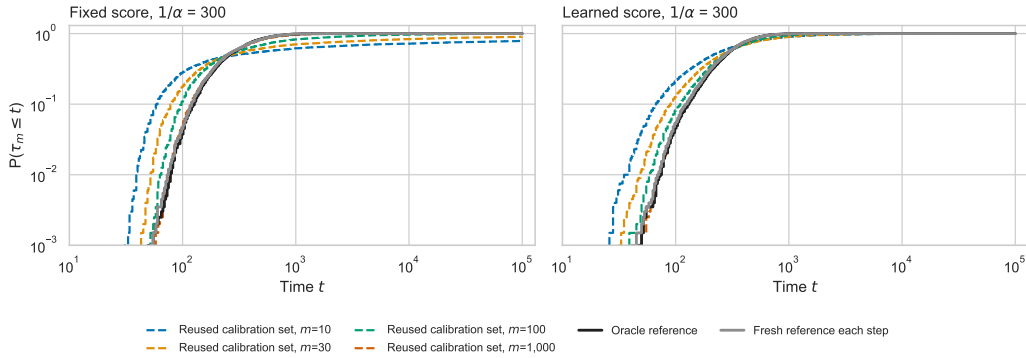
D.1 Evaluating ARL_0 control for varying calibration dataset sizes

To evaluate ARL_0 control for finite calibration dataset sizes m , we simulate null streams at varying values of m for two settings: a fixed versus learned witness function. In the fixed-score setting, scores $\hat{f}_t(X_t)$ and $\hat{f}_t(Z_i)$ are drawn i.i.d. from a Beta(2, 2) distribution. In the learned-score setting, the data X_t is drawn from a 5-dimensional standard normal distribution with the bettor continually refitting the changepoint and retraining a logistic classifier. The calibration dataset size m is varied across values $\{10, 30, 100, 1000\}$.

As shown in Figure D.1, the empirical ARL_0 is larger than the nominal ARL_0 in nearly every setting except for the smallest calibration dataset size of $m = 10$ and the largest target ARL_0 value of 10000. Interestingly, the empirical ARL_0 is extremely conservative for the setting with fixed witness functions, while the empirical ARL_0 is much closer to the nominal ARL_0 value for the learned witness. In addition, Figure D.2 compares the distribution of null alarm times at $1/\alpha = 300$, within the range of thresholds in our experiments, with that of the oracle detector, which uses the exact pre-change expectation. With $m = 30$, the distribution is similar to the oracle’s for both the fixed and the learned score, and smaller calibration datasets move some alarms earlier. Based on these results, our experiments use $m = 30$.

Table C.4: Concepts active at the end of the first four AI scribe streams in Figure 3(h), with the theme each is assigned in Table C.3 and the fraction of notes for which it is annotated as present before and after the update.

Stream	Concept	Theme	Before $\rightarrow$ after
1	Does the edit add or correct procedural or diagnostic details based on inaccuracies or omissions in the initial draft?	Clinical correction	0.56 $\rightarrow$ 0.67
	Does the edit clarify or correct medication, management, or demographic information based on inaccuracies or omissions in the initial draft?	Clinical correction	0.61 $\rightarrow$ 0.72
	Does the edit add or correct missing or inaccurate patient history and administrative details (such as functional status, anatomical location, provider identification, or lab data) in the initial draft?	Non-clinical correction	0.63 $\rightarrow$ 0.76
	Does the edit add or correct missing treatment and clinical history details based on inaccuracies or omissions in the initial draft?	Clinical correction	0.53 $\rightarrow$ 0.64
2	Does the edit remove unnecessary or irrelevant patient-related information (such as identification, introduction, background, history, or personal details) from the initial draft?	Unneeded content	0.82 $\rightarrow$ 0.66
	Does the edit improve procedural and clinical detail documentation by clarifying, correcting, or adding missing information based on the initial draft?	Clinical correction	0.53 $\rightarrow$ 0.72
	Does the edit improve the note content by adding, clarifying, condensing, or editing the clinical note content?	Clinical correction	0.64 $\rightarrow$ 0.84
	Does the edit remove patient identifiers or identifiable patient details from the initial draft?	Unneeded content	0.78 $\rightarrow$ 0.52
3	Does the initial draft include unnecessary or irrelevant non-clinical information such as review of systems, introductory sentences, or personal details?	Unneeded content	0.93 $\rightarrow$ 0.60
	Does the edit remove patient identifiers or sensitive non-clinical information?	Unneeded content	0.81 $\rightarrow$ 0.54
	Does the edit remove unnecessary, redundant, or irrelevant content?	Unneeded content	0.91 $\rightarrow$ 0.68
	Does the edit restructure, clarify, or remove unnecessary note structure or introductory content?	Unneeded content	0.95 $\rightarrow$ 0.73
4	Does the edit restructure the note and remove unnecessary introductory sentences or note sections from the initial draft?	Unneeded content	0.85 $\rightarrow$ 0.61
	Does the edit correct inaccurate information in the initial draft, such as medication orders, clinician names, or treatment details?	Clinical correction	0.54 $\rightarrow$ 0.74
	Does the edit remove personal or demographic sensitive information from the initial draft?	Unneeded content	0.79 $\rightarrow$ 0.54
	Does the edit remove patient identifiers or personal information from the initial draft?	Unneeded content	0.76 $\rightarrow$ 0.58


 Figure D.2: Distribution of the null alarm time τ_m at $1/\alpha = 300$ with a fixed score (left) and a learned score (right). Every null stream draws its own calibration dataset, so each curve estimates $P(\tau_m \leq t)$ averaged over calibration datasets from 2,000 null streams. The black curve is the oracle detector, which uses the exact pre-change expectation.

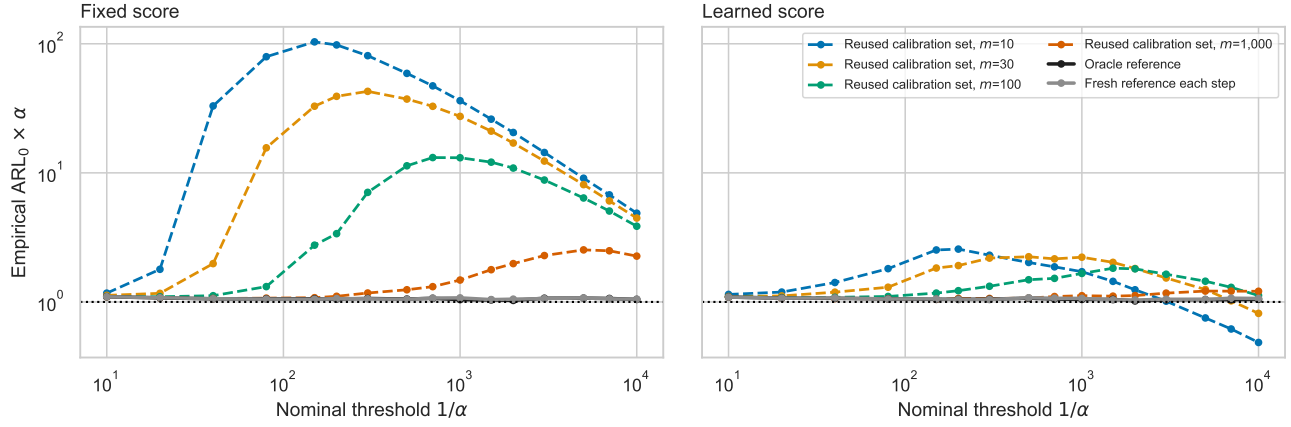


Figure D.1: Empirical ARL_0 as a multiple of the nominal $ARL_0 = 1/\alpha$ with a fixed score (left, Beta(2, 2) scores) and with a learned score from a classifier refit on the stream history (right). Dashed colored lines reuse a calibration dataset of size $m \in \{10, 30, 100, 1000\}$ at every step, the solid black line uses the oracle expectation $\mathbb{E}_{P_0}[\hat{f}_t(X)]$, and the solid grey line redraws 10 reference observations at every step. The dotted line marks the nominal ARL_0 . Each point summarizes 2,000 null streams, and streams without an alarm after 100,000 steps are counted at 100,000, so each value is a lower bound.