MISTAKE-BOUNDED LANGUAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the learning task of language generation in the limit, but shift focus from the traditional time-of-last-mistake metric of a generator's success to a new notion of *mistake-bounded generation*. While existing results for language generation in the limit focus on guaranteeing eventual consistency, they are blind to the cumulative error incurred during the learning process. We address this by shifting the goal to minimizing the total number of invalid elements output by a generation algorithm. We establish a formal reduction to the Learning from Correct Demonstrations framework of Joshi et al., enabling a general recipe for deriving mistake bounds via weighted update rules. For finite classes, we provide an algorithm that simultaneously achieves an optimal last-mistake time of $\mathsf{Cdim}(\mathcal{L}) + 1$ and a mistake bound of $\lfloor \log_2 |\mathcal{L}| \rfloor$, whereas for the non-uniform setting of countably infinite streams of languages, we prove a fundamental trade-off: achieving logarithmic mistakes $O(\log i)$ necessarily precludes convergence guarantees established in prior work. Finally, we show that our framework can be extended to accommodate noisy adversaries and guarantee mistake bounds that scale with the adversary's suboptimality.

1 INTRODUCTION

The capability of Large Language Models (LLMs) to generate coherent text has sparked a renewed interest in the theoretical underpinnings of language generation. While early learning-theoretic work such as Gold and Angluin's work on language identification in the limit (Gold, 1967; Angluin, 1980) focused on the difficult task of identifying a language or grammar from positive samples, recent work by Kleinberg & Mullainathan (2024) hinges on the observation that a successful large language model need not fully identify its language of interest, but simply understand enough to generate new, valid examples from the language.

This idea has been explored through the model of *language generation in the limit*, first proposed by Kleinberg & Mullainathan (2024). In this setting, a generator plays an online game against an adversary. At each step, the generator must output a new element $\hat{x}_t$ that has not yet been outputted by the adversary, and then the adversary reveals a new element x_t from a target language L^*, known to come from some class of languages $\mathcal{L}$. The generator succeeds if it eventually converges to outputting only elements from L^*. Subsequent work by Li et al. (2024) further refined what it means to *eventually converge*, dividing the space of language generation requirements into three tiers, listed in order of increasing strength: language generation in the limit, non-uniform generation, and uniform generation.

Existing work in all three of these settings quantifies a generator's success through the *time-of-last-mistake*, i.e. the timestep after which the generator always outputs from the target language L^*. While this metric guarantees eventual consistency, it is blind to the cost of learning incurred along the way. A generator could theoretically hallucinate invalid outputs for a million steps before converging, or could generate perfectly for a long time before making a single mistake after a million steps. In both cases, the "time of last mistake" might be identical, yet the practical utility of these two generators is vastly different. The following example shows that this latter case is indeed possible:

Example 1 (Separation between time-of-last mistake and number of mistakes.)**.** *Consider a language class composed of two languages $\mathcal{L} = \{L_1, L_2\}$, where $|L_1 \cap L_2| = n < \infty$. Consider the adversary that enumerates the elements falling in $L_1 \cap L_2$ for the first n timesteps. At timestep $t = n + 1$, there are no unseen elements remaining in the intersection, and thus the generator is forced to output an element in $L_1 \setminus L_2$ or $L_2 \setminus L_1$. In either case, the adversary can force a mistake by observing the*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

generator's choice, and then enumerating elements from the other language. Thus, the adversary can force the learner's time of last mistake to be as large as $n + 1$.

In contrast, if the generator also enumerates $L_1 \cap L_2$ while there are still unseen elements in the intersection, the generator is guaranteed to only make a single mistake at time $t = n + 1$ when it must play the first point outside of the intersection.

Clearly the total number of mistakes is at most the time of the last mistake (since the generator outputs at most one element per time step), but Example 1 shows that the total number of mistakes can be arbitrarily lower: in the example, the time of the last mistake can be arbitrarily high depending on the choice of language class, but the number of mistakes is always bounded by 1. The fact that these two performance metrics operate in different ways raises a fundamental set of questions analogous to the study of mistake bounds in online classification (Littlestone, 1988):

> **Q1.** *Are there upper bounds on the total number of mistakes a generator makes for general language classes that are dramatically better than the trivial upper bound inherited from the time of the last mistake?*
>
> **Q2.** *Does minimizing the total number of mistakes a generator makes require sacrificing optimal guarantees on the time when it makes its last mistake?*

We investigate this question by studying the interplay between the two metrics of last-mistake-time and total mistakes. While bounding mistakes is valuable on its own, an ideal generator would provide a dual guarantee: minimizing the total volume of errors while also ensuring that the generator converges to the true language as quickly as possible.

1.1 Our Contributions

In this work, we introduce the framework of *Mistake-Bounded Language Generation*. We demonstrate that while the total number of mistakes is bounded by the time-of-last-mistake, it can be arbitrarily smaller. We provide algorithms that minimize this new metric and prove fundamental trade-offs between minimizing errors and minimizing convergence time.

Our primary contributions are:

1. **A General Recipe for Mistake-Bounded Generation via Learning from Demonstrations:** We prove bounds on the total number of mistakes for language generation by first establishing a formal connection between language generation and the setting of Learning from Correct Demonstrations (LfD) (Joshi et al., 2025). Leveraging this reduction, we adapt their techniques to provide mistake bounds in the language generation setting. Crucially, while the original LfD technique is restricted to finite classes, we extend the analysis to handle *countably infinite* language streams through the use of a growth function and non-uniform prior over languages.
2. **Optimal Bounds for Finite Classes:** For finite language classes, we prove an upper bound of $\min\{\log_2 |\mathcal{L}|, \mathsf{Cdim}(\mathcal{L}) + 1\}$ on the total mistakes. This directly addresses (Q1) above for the finite class case: While the closure dimension has been shown to tightly govern the optimal time-of-last-mistake for finite classes, our bound shows that we can do dramatically better in terms of total mistakes when $\log |\mathcal{L}|$ is much smaller than $\mathsf{Cdim}(\mathcal{L})$. Furthermore, in response to (Q2) as posed above, we present a hybrid algorithm that *simultaneously* achieves this mistake bound and the optimal time-of-last-mistake, proving that one need not sacrifice convergence speed to minimize errors in the finite class setting.
3. **Non-Uniform Bounds and Trade-offs:** In Section 6, we apply our general recipe to infinite language classes $\mathcal{L} = \{L_1, L_2, \dots\}$, deriving a non-uniform mistake bound of $O(\log i)$ for the i-th language. This provides a positive answer to (Q1) for the non-uniform setting, but reveals a more nuanced answer to (Q2) than the finite case: here, minimizing mistakes *does* require sacrificing bounds on the time-of-last-mistake. We formalize this via a fundamental trade-off (Theorem 6.3), proving that any algorithm achieving the non-uniform convergence time guarantees of (Charikar & Pabbaraju, 2024) must suffer a number of mistakes *linear* in i, whereas our algorithm achieves a logarithmic bound by allowing for a slower convergence time.

4. **Robustness to Noise:** Finally, we show that our reduction to the LfD setting naturally handles adversarial noise. We give mistake bounds for finite and countably infinite language classes in Lemmas 7.1 and 7.2, respectively, and contrast these results with prior work on last-mistake-time guarantees in the presence of noise.

2 Related Work

Building on Kleinberg & Mullainathan (2024), subsequent work has advanced our understanding of language generation in the limit and investigated various extensions of the model. Li et al. (2024) introduce and study the stronger models of non-uniform and uniform generation, and give characterizations of when these stronger generation goals are possible. Charikar & Pabbaraju (2024) also give a simple algorithm for non-uniform generation that matches and in some cases improves the guarantees of Li et al. (2024), while exploring a number of other variants to the language generation model.

Other follow-up works have sought to further understand which classes are and are not generatable. Charikar & Pabbaraju (2025) study the pareto-frontier of non-uniform generation guarantees, while Hanneke et al. (2025) study whether finite unions of generatable classes are guaranteed to satisfy generatability, and answer this in the negative. Anastasopoulos et al. (2026) study a variant termed *safe* language generation, in which a generator must output elements from the target language while avoiding a subset of harmful elements. Karbasi et al. (2025) consider the problem of auditing a generator to determine whether it is hallucinating. Arenas et al. (2025) study the problem of language generation through a computational lens, and show barriers to computationally-efficient generation in various settings.

A significant line of works have considered the possibilities and impossibilities around *language generation with breadth*, in which a generator must not only output consistent elements, but guarantee eventual "coverage" of the target language according to various definitions Kalavasis et al. (2024; 2025); Charikar & Pabbaraju (2024); Peale et al. (2025); Kleinberg & Wei (2025a;b).

Finally, a last variant that has received significant study is language generation in the presence of *noisy* adversaries. We overview these works in Section 7.

3 Mistake-Bounded Language Generation

We begin by overviewing the learning task of language generation in the limit, introduced by Kleinberg & Mullainathan (2024). We will focus on the settings of uniform and non-uniform language generation by Li et al. (2024). We then define our new notion of *mistake-bounded generation*, and how it fits into this existing framework.

Given a countably infinite universe $\mathcal{X}$, a language generation in the limit problem is defined with respect to a (potentially infinite) language class $\mathcal{L}$, where each language $L \in \mathcal{L}$ specifies a particular infinite subset of the universe $\mathcal{X}$. When $\mathcal{L}$ is a countably infinite class, we assume that it is provided to us in the form of an infinite stream of languages $L_1, L_2, ...$

The learning task of language generation in the limit is formulated as an online game between a generation algorithm and an adversary. At each timestep t, the generator outputs an element $\hat{x}_t \in \mathcal{X}$, and then the adversary reveals an element $x_t \in \mathcal{X}$ (See Remark 1 for a note on order-of-play). At each timestep, the generator must operate under the constraint that $\hat{x}_t$ cannot be an element that has been previously outputted by the adversary, e.g. $\hat{x}_t \notin \{x_1, ..., x_{t-1}\}$.

We make the following assumptions about the adversary. First, we assume that the adversary's outputs realize some language $L^* \in \mathcal{L}$, i.e. there exists an $L^* \in \mathcal{L}$ such that $x_t \in L^*$ for all timesteps $t \in \mathbb{N}$. Second, we assume that each of the adversary's outputs are unique, i.e. $x_t \notin \{x_1, ..., x_{t-1}\}$. See Remark 1 for further discussion about this assumption.

At each timestep t, the generator gets no information about the identity of the true language L^* beyond the transcript of elements outputted by the adversary thus far, $x_1, ..., x_{t-1}$, which we will denote as $x_{1:t-1}$. Nevertheless, the generator is judged by its ability to eventually generate elements from L^*. The particular notion of *eventually* is what separates the three tiers of language generation,

listed in order of strength: language generation in the limit, non-uniform generation, and uniform generation. In this paper, we focus on the latter two, and define them below.

Definition 1 (Non-Uniform Language Generation)**.** *A generation algorithm $\mathcal{G}$ satisfies non-uniform generation for a language class $\mathcal{L}$ if for every $L \in \mathcal{L}$, there exists a finite $t(\mathcal{G}, L) < \infty$ such that, when given as input any stream of unique adversarial outputs $x_1, x_2, ...$ realized by L, for all timesteps $t \geq t(\mathcal{G}, L)$, the algorithm is guaranteed to generate from L, i.e. $\hat{x}_t \in L \setminus \{x_{1:t-1}\}$.*

In non-uniform generation, the generator is guaranteed to start generating elements consistent with the true language at some finite timestep. However, the particular timestep where this is guaranteed to happen can be different for each language. This means that it's possible that even though $t(\mathcal{G}, L)$ is finite for each language, $\max_{L \in \mathcal{L}} t(\mathcal{G}, L)$ is infinite. Uniform generation strengthens the requirement to hold over all languages. In other words, the value of this max expression must be finite:

Definition 2 (Uniform Language Generation)**.** *A generation algorithm $\mathcal{G}$ satisfies uniform generation for a language class $\mathcal{L}$ if there exists a time $T \in \mathbb{N}$ such that for adversarial outputs $x_1, x_2, ...$ realized by a language $L \in \mathcal{L}$, for all timesteps $t \geq T$ the generator is guaranteed to generate from L, i.e. $\hat{x}_t \in L \setminus \{x_{1:t-1}\}$.*

Remark 1. *We note two deviations from standard generation setups, both made purely for expositional ease. First, we reverse the standard order of play so that the generator moves first. This choice does not materially affect theoretical guarantees, as any algorithm designed for one setting can be adapted to the other, shifting the total mistake bound or time-of-last-mistake by at most 1. Second, we assume the adversary outputs only unique elements. Existing definitions already measure eventual consistency times ($t(\mathcal{G}, L)$ and T) against the number of* unique *items seen, preventing the adversary from stalling learning by indefinitely repeating elements. Our setup simply ignores steps where a duplicate is output.*

3.1 A NEW SUCCESS METRIC: NUMBER OF MISTAKES

One can view the guarantees of uniform and non-uniform generation as providing a bound on the *time-of-last-mistake*, i.e., the last timestep t where the generator outputs an element that is not part of the true language. As illustrated in Example 1, the *total number of mistakes* made by the generator can be significantly smaller than the time of last mistake.

In this work, we aim to minimize this cumulative error. While for finite classes we simply aim to bound the total mistakes in terms of $|\mathcal{L}|$, for the countably infinite classes considered in Section 6, we require a definition that allows the mistake bound to scale with the complexity (position) of the target language in the stream:

Definition 3 (Non-Uniform Mistake Bound)**.** *Given a countably infinite language class equipped with a fixed enumeration $\mathcal{L} = \{L_1, L_2, ...\}$, a generator $\mathcal{G}$ and a target language L_i, $\mathcal{G}$'s* non-uniform mistake bound *with respect to L_i is defined as the worst-case number of mistakes over all possible streams of unique elements from L_i:*

$$M_{\text{non-unif}}(\mathcal{G}, i) := \sup_{x_1, x_2, \cdots \in L_i} \sum_{t=1}^{\infty} \mathbf{1}\big[\mathcal{G}(x_{1:t-1}) \notin L_i\big].$$

Thus, the goal for infinite language classes is to provide a finite bound on $M_{\text{non-unif}}(\mathcal{G}, i)$ for every language $L_i \in \mathcal{L}$.

4 A GENERAL RECIPE FOR MISTAKE-BOUNDED GENERATION

While prior algorithms for language generation in the limit rely on keeping track of which languages are still consistent with the adversarial stream, bounding the total number of mistakes requires a refined approach that not only keeps track of which languages are consistent, but prioritizes generating from languages on which the generator has already made many mistakes.

Instead, we adopt a *weighted* approach derived from a reduction to the "Learning from Correct Demonstrations" (LfD) framework introduced by Joshi et al. (2025). In Appendix E.3, we show how the problem of language generation can be mapped to the LfD setting.

Our core algorithmic strategy builds upon the multiplicative weights update technique proposed by Joshi et al. (2025) for the LfD setting. However, a direct application of their result is insufficient due to the fact that their result only implies mistake bound for finite language classes. The setting of language generation, particularly non-uniform generation, inherently requires handling *countably infinite* streams of languages.

To bridge this gap, we introduce Algorithm 1, which extends the ideas of Joshi et al. (2025) to infinite language classes. This extension requires two new components not present in the original LfD setting:

1. A **prior weight distribution** $w_0 : \mathbb{N} \to \mathbb{R}_{>0}$, representing our initial belief in the likelihood of each language.
2. A **growth function** $f : \mathbb{N} \to \mathbb{N}$, which governs the "capacity" of the learner by determining how many languages are actively considered at timestep t. This growth function allows the algorithm to handle infinite streams with finite computational resources.

Remark 2. *While we present this algorithm in the context of language generation, this extension also yields a novel result for the original LfD setting: it provides the first mistake bounds for learning from infinite streams of reward functions (see Appendix E.3).*

We show that our algorithm yields the following mistake bound guarantee:

Theorem 4.1. *For any countably infinite language class equipped with a fixed enumeration $\mathcal{L} = \{L_1, L_2, ...\}$, associated weights $w_0 : \mathbb{N} \to \mathbb{R}_{>0}$ such that $\sum_{i=1}^{\infty} w_0(i) \leq W < \infty$, and non-decreasing growth function $f : \mathbb{N} \to \mathbb{N}$, the generations outputted by Algorithm 1 guarantee a non-uniform mistake bound of $M_{\text{non-unif}}(\mathcal{G}, i) \leq f^{-1}(i) + \lfloor \log_2(W/w_0(i)) \rfloor$ for each $i \in \mathbb{N}$, where if there exists a t' such that $f(t') \geq i$, $f^{-1}(i)$ is the largest value t such that $f(t) < i$, or 0 if for every $t' \in \mathbb{N}$ $f(t')$ is at least i, and otherwise $f^{-1}(i) = \infty$.*

The proof (provided in Appendix A) relies on a potential function argument tracking the total weight of the system. Intuitively, every mistake made while L_i is the target causes $w_t(i)$ to double. However, we also show that the actions taken by the generator ensure that the total weight of the system is bounded, thus preventing $w_t(i)$ from doubling indefinitely.

In the following sections, we instantiate this recipe with different weights w_0 and growth functions f to derive mistake bounds for both finite (Section 5) and infinite (Section 6) classes.

Algorithm 1 Non-Uniform Mistake-Bounded Generation algorithm for a provided initial weighting of languages and growth function.

Input: Universe $\mathcal{X}$, class $\mathcal{L} = \{L_1, L_2, \dots\}$, weights $w_0 : \mathbb{N} \to \mathbb{R}_{\geq 0}$, growth function $f : \mathbb{N} \to \mathbb{N}$
for $t = 1, \dots, \infty$ **do**
Output $\hat{x}_t := \arg\max_{x \in \mathcal{X} \setminus x_{1:t-1}} \sum_{i=1}^{f(t)} w_{t-1}(i)\mathbf{1}[x \in L_i]$
Observe adversary's x_t
Update weights for $i \in [f(t)]$:

$$w_t(i) = \begin{cases} 0 & x_t \notin L_i \\ w_{t-1}(i) & x_t, \hat{x}_t \in L_i \\ 2w_{t-1}(i) & x_t \in L_i, \hat{x}_t \notin L_i \end{cases}$$

Initialize weights for $i \in \{f(t)+1, \dots, f(t+1)\}$:

$$w_t(i) = \begin{cases} w_0(i) & x_{1:t} \subseteq L_i \\ 0 & \text{otherwise} \end{cases}$$

end

5 Generating from Finite Language Classes with $\log_2 |\mathcal{L}|$ mistakes

We begin by focusing on the special case of finite language classes. Existing results by Li et al. (2024) and Kleinberg & Mullainathan (2024) have shown that any finite language class is uniformly

generatable. Moreover, they prove that the optimal *time-of-last-mistake* is tightly characterized by the *closure dimension*.

Definition 4. *The* closure dimension *of a language class $\mathcal{L}$ is the size of the largest finite intersection of a sub-collection of languages from $\mathcal{L}$:*

$$\mathsf{Cdim}(\mathcal{L}) = \max\{\left|\bigcap_{L\in\mathcal{L}'} L\right| : \mathcal{L}' \subseteq \mathcal{L}, \left|\bigcap_{L\in\mathcal{L}'} L\right| < \infty\}.$$

If no such finite intersection exists, the closure dimension is defined to be 0. If there is no finite maximum, the dimension is defined to be ∞.

While Li et al. (2024) provide an algorithm that guarantees that the last mistake happens by time $t = \mathsf{Cdim}(\mathcal{L}) + 1$, they also show that this bound is tight: there exist adversarial streams that force the generator to make a mistake at $t = \mathsf{Cdim}(\mathcal{L}) + 1$. However, this metric does not consider the total number of mistakes made on the way to the time of last mistake.

5.1 A Hybrid Algorithm: Minimizing Mistakes and Convergence Time

We now show that by instantiating our general recipe for mistake-bounded language generation (Algorithm 1) with uniform weights, we can achieve a mistake bound that can be arbitrarily smaller than the closure dimension. Furthermore, we show that our algorithm's decisions coincide with critical decision points in the uniform generation algorithm of Li et al. (2024), thus satisfying *both* optimality criteria simultaneously.

Let $\mathcal{L}$ be a finite class of size N. We instantiate Algorithm 1 with uniform weights $w_0(i) = 1$ for all $L_i \in \mathcal{L}$ and a static growth function that considers all languages at every point in time: $f(t) = N$ for all t. While this algorithm alone only provides a mistake bound, we show that its actions also satisfy a critical property: whenever the intersection of all consistent languages has infinite size, it plays an unseen element from this intersection. This is exactly the property required by the uniform generation algorithm of Li et al. (2024), and thus we can show that our algorithm also enjoys a last-mistake-time guarantee of $\mathsf{Cdim}(\mathcal{L}) + 1$.

This analysis yields the following dual guarantee:

Theorem 5.1. *For any finite class $\mathcal{L}$, there exists a generation algorithm that uniformly generates from $\mathcal{L}$ and guarantees that for any target $L^* \in \mathcal{L}$ and adversarial sequence consistent with L^*:*

1. *The total number of mistakes is at most* $\min\{\lfloor \log_2 |\mathcal{L}| \rfloor, \mathsf{Cdim}(\mathcal{L}) + 1\}$.
2. *The time of last mistake is at most* $\mathsf{Cdim}(\mathcal{L}) + 1$.

The full proof can be found in Appendix B.1. This result demonstrates a fundamental separation between our two metrics of total mistakes and time-of-last-mistake. This result recovers the intuition of Example 1, in which the time until the last mistake depended on the closure dimension, $|L_1 \cap L_2|$, but there was a strategy that preserved this time-of-last-mistake guarantee while only making $\log_2 |\mathcal{L}| = 1$ mistake.

5.2 Lower Bounds

We note that while our algorithm's last mistake bound has been shown to be tight for every class, the mistake bound is not necessarily optimal in a per-class sense. However, the following lower bound demonstrates that this is the best we can hope to achieve from a bound that depends solely on the cardinality of $\mathcal{L}$. We prove this in Appendix B.2 via a lower bound construction based on Littlestone trees (see Figure 1).

Lemma 5.2 (Lower Bound)**.** *For any $N \in \mathbb{N}$, there exists a finite language class $\mathcal{L}$ of size N such that no generation algorithm can guarantee fewer than $\lfloor \log_2 N \rfloor$ mistakes on every adversarial stream consistent with an $L \in \mathcal{L}$.*

6 MISTAKE-BOUNDED NON-UNIFORM GENERATION

We now turn to the general setting of countably infinite language classes $\mathcal{L} = \{L_1, L_2, \dots\}$. Here, we cannot hope for a uniform bound on mistakes across the entire class. Instead, we seek a *non-uniform* guarantee that depends on the index i of the target language L_i. Existing work by Li et al. (2024) and Charikar & Pabbaraju (2024) focuses on minimizing the *time of last mistake*. For example, the Greedy algorithm of Charikar & Pabbaraju (2024) guarantees convergence in terms of the *non-uniform complexity of* L_i, $m(L_i)$, a structural parameter representing the complexity of L_i's intersections with preceding languages.

Definition 5 (Non-Uniform Complexity)**.** *Given an infinite stream of languages $\mathcal{L} = L_1, L_2, ...,$ the* non-uniform complexity *of L_i is defined as size of the largest finite intersection of a subcollection of $(L_1, ..., L_i)$ that contains L_i:*

$$m(L_i) := \max_{\mathcal{L}' \subseteq \{L_1, \dots, L_{i-1}\}} \{ | \bigcap_{L \in \mathcal{L}'} L \cap L_i | : | \bigcap_{L \in \mathcal{L}'} L \cap L_i | < \infty \}.$$

We define $m(L_i) = 0$ if no such finite intersection exists.

In this section, we present two contrasting results. First, we show that if one prioritizes minimizing mistakes, we can achieve a mistake bound of $O(\log_2 i)$, i.e. with only logarithmic dependence on the index. However, we then prove that this comes at a cost: no algorithm can simultaneously achieve this logarithmic mistake bound and the convergence time guaranteed by Charikar & Pabbaraju (2024).

6.1 MINIMIZING MISTAKES WITHOUT CONVERGENCE

To minimize the total number of mistakes, we revisit our General Recipe (Algorithm 1). In the finite case, we used a uniform prior. For infinite streams, we must adopt a prior that decays sufficiently fast to ensure finite capacity, yet slowly enough to not penalize later languages too heavily. We instantiate Algorithm 1 with polynomially decaying weights $w_0(i) = \frac{1}{i^2}$ (noting that $\sum \frac{1}{i^2} < \infty$). and a growth function that doubles the number of languages considered at each step, $f(t) = 2^t$.

Under this instantiation, our algorithm yields the following guarantee:

Theorem 6.1. *Let $\mathcal{L} = \{L_1, L_2, ...\}$ be a countably infinite class of languages under a fixed enumeration. For any language $L_i \in \mathcal{L}$, the instantiation of Algorithm 1 described above guarantees a non-uniform mistake bound of:*

$$M_{\text{non-unif}}(\mathcal{G}, i) \leq 3 \log_2 i + \log_2(\pi^2/6) = O(\log_2 i).$$

The proof follows immediately from an appropriate instantiation of Theorem 4.1 (see Section C.1).

6.2 SIMULTANEOUSLY BOUNDING MISTAKES AND CONVERGENCE

While the strategy above guarantees low error, it does not prioritize *when* those errors occur. In particular, by prioritizing bounding mistakes on languages late in the stream, it may be slow to converge on early languages.

We next present an algorithm that can non-uniformly generate from $\mathcal{L}$ while simultaneously bounding the total number of mistakes. However, in order to balance these two goals, we must weaken our mistake bound to only guaranteeing a mistake bound that is linear in the index of the target language. To obtain this guarantee we analyze a modification of the greedy non-uniform generation algorithm by Charikar & Pabbaraju (2024), termed MODIFIED-GREEDY. This algorithm prioritizes maintaining consistency with the earliest consistent languages.

Lemma 6.2. *There exists a non-uniform generation algorithm (*MODIFIED-GREEDY*) with a time-of-last-mistake guarantee of $\max\{i-1, m(L_i)+1\}$ and a non-uniform mistake bound of:*

$$M_{\text{non-unif}}(\mathcal{G}, i) \leq \min\{2(i-1), \max\{i-1, m(L_i)+1\}\}.$$

Note that the mistake bound here is linear in i ($2(i-1)$), which is exponentially worse than the $O(\log i)$ bound in Theorem 6.1. We present the proof in Appendix D.

6.3 Tradeoff Between Last-Mistake-Time and Total Mistakes

This raises a natural question: is there a generation strategy that achieves our optimal mistake bound of $O(\log_2 i)$ while also preserving Charikar & Pabbaraju (2024)'s last-mistake-time guarantee?

We answer in the negative. We prove a fundamental barrier: to achieve the optimal convergence time, an algorithm must be willing to make mistakes on later languages in order to prioritize converging on languages appearing earlier in the stream. Conversely, to achieve logarithmic mistakes, an algorithm must start to prioritize later languages over early languages when they start to accumulate many mistakes, potentially delaying convergence time.

Theorem 6.3. *There exists an infinite language class with a fixed enumeration $\mathcal{L} = \{L_1, L_2, \ldots\}$ such that, for every generation algorithm $\mathcal{G}$ satisfying*

$$t_{\mathcal{G}}(L_i) \leq C_{\mathcal{G}} \max\{i, m(L_i)\}$$

for some finite constant $C_{\mathcal{G}}$ and every i, there exists a constant $K_{\mathcal{G}}$ such that

$$M_{\text{non-unif}}(\mathcal{G}, i) \geq i - K_{\mathcal{G}}$$

for every $i > K_{\mathcal{G}}$. In particular,

$$M_{\text{non-unif}}(\mathcal{G}, i) = \Omega(i).$$

The full proof of this result can be found in Appendix D.1. The key intuition is as follows: we construct a sequence of languages sharing nested, rapidly growing finite prefixes. By sequentially enumerating these shared elements, the adversary forces the generator into a recurring dilemma at each prefix boundary. At the kth boundary, the unseen points available to the generator are either points solely in the language L_k, or a point shared by all subsequent languages L_j for $j \geq k$, but outside of L_k. If the generator predicts a point outside of L_k, the adversary halts and declares that language the target. This forces a late mistake that violates the time-of-last-mistake bound for L_k. On the other hand, if the generator safely plays from L_k at each boundary, it means that by the time it gets to the ith boundary, it has incurred a mistake on L_i at every boundary leading up to i, resulting in a linear number mistakes if the adversary halts and declares L_i the target.

7 Mistake-Bounded Noisy Generation

The algorithms presented in Sections 5 and 6 rely on eliminating candidate languages as soon as they become inconsistent with the adversary's outputs. While this approach works well in a setting where we assume the adversary is perfectly consistent with a target language, it's also natural to explore what guarantees are possible against *noisy* adversaries, who may not be perfectly consistent with the target language.

This imperfect setting has been studied in the context of last-mistake-time guarantees by a number of works. Raman & Raman (2025) first proposed a noisy setting where the adversary may insert noise at an unknown but finite number of steps. More recently, Bai et al. (2025) further explore the limits of generating in the presence of a finite amount of noise, while Mehrotra et al. (2025) show a surprising result that any countable class can be generated in the limit in the presence of potentially infinite but vanishing noise.

In this section, we show that noise-dependent mistake bounds for the noisy setting can be derived by leveraging results from the learning from demonstrations setting. The key idea behind these results is to relax the weight updates to downweight languages when the adversary makes a mistake, rather than setting the weight to 0 and removing them from consideration.

Lemma 7.1 (Noisy Mistake Bound for Finite Classes)**.** *Given a finite language class $\mathcal{L}$ and $\gamma \in (0, 3/4]$, given an adversarial stream $x_1, x_2, \ldots$, not necessarily consistent with any $L \in \mathcal{L}$, denote the number of mistakes the stream has made on L at time t as*

$$M_L(x_{1:t}) = \sum_{i=1}^{t} \mathbf{1}[x_i \notin L].$$

Similarly, denote the learner's mistakes up to time t as $M_L(\hat{x}_{1:t})$. Then, there exists an algorithm that guarantees for any L,

$$M_L(\hat{x}_{1:t}) \leq (1 + 2\gamma) M_L(x_{1:t}) + \log_2 |\mathcal{L}|/\gamma.$$

This noisy guarantee follows from the results of Joshi et al. (2025) on learning from demonstrations. In Appendix E.2, we show its derivation. If we instantiate this result with $\gamma = 1/2$ and assume our adversary is guaranteed to only have a finite amount of noise, then this result guarantees that our generator will only make a finite number of mistakes, no matter the amount of noise. In particular, its mistakes are bounded by $2(M + \log_2 |\mathcal{L}|)$, where M is the amount of noise inserted by the adversary.

While the results of Joshi et al. (2025) do not extend to infinite streams of reward functions, we provide an extension of their result (Theorem E.5) that allows us to derive non-uniform mistake bounds for infinite streams of languages. Proper instantiation of the theorem's parameters gives the following guarantee for language generation in the presence of noisy adversaries (see Appendix E.2 for details):

Lemma 7.2 (Noisy Mistake Bound for Infinite Streams)**.** *Suppose we are given an infinite stream of languages $\mathcal{L} = L_1, L_2, ...$ and $\gamma \in (0, 3/4]$. Then, there exists a generation algorithm such that for every adversarial stream $x_1, x_2, ...$, not necessarily consistent with any $L_i \in \mathcal{L}$ and $i, T \in \mathbb{N}$,*

$$\sum_{t=1}^{T} \mathbf{1}[\hat{x}_t \notin L_i] \leq (1 + 2\gamma) \sum_{t=\max\{\lfloor \log_2 i \rfloor, 1\}}^{T} \mathbf{1}[x_t \notin L_i] + (1 + 2/\gamma) \log_2 i + \frac{\log_2(\pi^2/6)}{\gamma}.$$

As in the finite case, we note that this result implies that when the adversary's stream only has a finite amount of noise, the generator is also guaranteed to make at most a finite number of mistakes.

7.1 Comparison to existing time-of-last-mistake bounds

The guarantee of Lemma 7.1 echoes that of Raman & Raman (2025), who show that whenever the adversary only adds a finite amount of noise, the time-of-last-mistake can be bounded for any finite language class. Similarly, Lemma 7.2 guarantees in cases where the adversary makes a finite amount of mistakes, we can guarantee a non-uniform but finite number of mistakes for any countable class. This can be compared to the results of Mehrotra et al. (2025), who show that any countable language can be generated in the limit in the presence of a finite amount of noise. We leave as a key open question whether the time-of-last-mistake bounds given by Raman & Raman (2025) and Mehrotra et al. (2025) can be achieved simultaneously with the mistake bounds guaranteed in Lemmas 7.1 and 7.2, respectively.

Interestingly, while Mehrotra et al. (2025) are able to guarantee a finite number of mistakes (upper-bounded by the finite last-mistake-time) in the presence of infinite but vanishing noise, our mistake bounds do not necessarily guarantee a finite number of mistakes in such a case, just that the rate of mistakes made by the learner is no worse than that of the adversary. We leave developing improved algorithms with a bounded number of mistakes in the $o(1)$-noise case, along with further analysis of the interaction between time-of-last-mistake and number of mistakes, as an open question for future work.

8 Open Directions

Our introduction of mistake bounds as an alternative metric of success in language generation raises several key questions. We note three of particular interest. (1) Our bounds for finite classes rely on cardinality ($\log_2 |\mathcal{L}|$). In online classification, the *Littlestone dimension* characterizes learnability independent of class size (Littlestone, 1988). Does there exist an analogous combinatorial dimension $d(\mathcal{L})$ for language generation that characterizes the optimal mistake bound more tightly than cardinality or closure dimension? [1] (2) Theorem 6.3 identifies a conflict between minimizing mistakes ($O(\log i)$) and minimizing time-of-last-mistake ($O(m(L_i))$). What is the full Pareto frontier of simultaneous mistake and convergence guarantees? (3) Finally, in the noisy setting, it remains an open problem whether our mistake bounds can be achieved *simultaneously* with bounded time-of-last-mistake guarantees.

[1] Concurrent work by Pour et al. (2026) suggests such a combinatorial dimension for the related problem of learning from demonstrations.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs, provide critical ingredients for proving mathematical claims, and formulate mathematical claims. I.e., brainstorming approaches, checking correctness and typos, simplifying arguments, and connecting techniques to existing work. We have not used generative AI tools to help develop theoretical models or conceptual frameworks or propose or refine hypotheses. Generating synthetic data sets, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, and interpreting results are not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures or images, summarize or analyse existing literature, brainstorm, source/search for information, edit the paper to improve readability, and identify relevant literature.

We have reviewed all AI-assisted work. All proofs and edits were independently checked and verified by the authors, and a manual literature search was conducted. All citations were added by hand after careful review of the cited source.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken care to clearly state all assumptions underlying our theoretical results, and complete proofs of all theorems and lemmas are provided in the appendix.

REFERENCES

Antonios Anastasopoulos, Giuseppe Ateniese, and Evgenios M Kornaropoulos. Safe language generation in the limit. *arXiv preprint arXiv:2601.08648*, 2026.

Dana Angluin. Inductive inference of formal languages from positive data. *Information and control*, 45(2):117–135, 1980.

Marcelo Arenas, Pablo Barceló, Luis Cofré, and Alexander Kozachinskiy. Language generation: Complexity barriers and implications for learning. *arXiv preprint arXiv:2511.05759*, 2025.

Yannan Bai, Debmalya Panigrahi, and Ian Zhang. Language generation in the limit: Noise, loss, and feedback. *arXiv preprint arXiv:2507.15319*, 2025.

Moses Charikar and Chirag Pabbaraju. Exploring facets of language generation in the limit. *arXiv preprint arXiv:2411.15364*, 2024.

Moses Charikar and Chirag Pabbaraju. Pareto-optimal non-uniform language generation. *arXiv preprint arXiv:2510.02795*, 2025.

Leonhard Euler. De summis serierum reciprocarum. *Commentarii academiae scientiarum Petropolitanae*, pp. 123–134, 1740.

E Mark Gold. Language identification in the limit. *Information and control*, 10(5):447–474, 1967.

Steve Hanneke, Amin Karbasi, Anay Mehrotra, and Grigoris Velegkas. On union-closedness of language generation. *arXiv preprint arXiv:2506.18642*, 2025.

Nirmit Joshi, Gene Li, Siddharth Bhandari, Shiva Prasad Kasiviswanathan, Cong Ma, and Nathan Srebro. Learning to answer from correct demonstrations. *arXiv preprint arXiv:2510.15464*, 2025.

Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. Characterizations of language generation with breadth. *arXiv preprint arXiv:2412.18530*, pp. 3, 2024.

Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. On the limits of language generation: Trade-offs between hallucination and mode-collapse. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pp. 1732–1743, 2025.

Amin Karbasi, Omar Montasser, John Sous, and Grigoris Velegkas. (im) possibility of automated hallucination detection in large language models. *arXiv preprint arXiv:2504.17004*, 2025.

Jon Kleinberg and Sendhil Mullainathan. Language generation in the limit. *Advances in Neural Information Processing Systems*, 37:66058–66079, 2024.

Jon Kleinberg and Fan Wei. Density measures for language generation. *arXiv preprint arXiv:2504.14370*, 2025a.

Jon Kleinberg and Fan Wei. Language generation and identification from partial enumeration: Tight density bounds and topological characterizations. *arXiv preprint arXiv:2511.05295*, 2025b.

Jiaxun Li, Vinod Raman, and Ambuj Tewari. Generation through the lens of learning theory. *arXiv preprint arXiv:2410.13714*, 2024.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

Anay Mehrotra, Grigoris Velegkas, Xifan Yu, and Felix Zhou. Language generation with infinite contamination. *arXiv preprint arXiv:2511.07417*, 2025.

Charlotte Peale, Vinod Raman, and Omer Reingold. Representative language generation. *arXiv preprint arXiv:2505.21819*, 2025.

Alireza F Pour, Farnam Mansouri, and Shai Ben-David. Learning with multiple correct answers–regret bounds under different feedback models. *arXiv preprint arXiv:2602.09402*, 2026.

Ananth Raman and Vinod Raman. Generation from noisy examples. *arXiv preprint arXiv:2501.04179*, 2025.

A Proofs from Section 4

In this section, we prove Theorem 4.1, which shows that our general algorithm (Algorithm 1) guarantees a mistake bound that depends on the choice of initial weights w_0 and growth function f. We restate the theorem here for readability:

Theorem 4.1. *For any countably infinite language class equipped with a fixed enumeration $\mathcal{L} = \{L_1, L_2, ...\}$, associated weights $w_0 : \mathbb{N} \to \mathbb{R}_{>0}$ such that $\sum_{i=1}^{\infty} w_0(i) \leq W < \infty$, and non-decreasing growth function $f : \mathbb{N} \to \mathbb{N}$, the generations outputted by Algorithm 1 guarantee a non-uniform mistake bound of $M_{\text{non-unif}}(\mathcal{G}, i) \leq f^{-1}(i) + \lfloor \log_2(W/w_0(i)) \rfloor$ for each $i \in \mathbb{N}$, where if there exists a t' such that $f(t') \geq i$, $f^{-1}(i)$ is the largest value t such that $f(t) < i$, or 0 if for every $t' \in \mathbb{N}$ $f(t')$ is at least i, and otherwise $f^{-1}(i) = \infty$.*

Our proof relies on the following lemma, which establishes a key property about how the weights in Algorithm 1 evolve:

Lemma A.1. *Define $W_t := \sum_{i=1}^{f(t+1)} w_t(i)$. Then, for any step $t \geq 1$,*

$$W_t \leq W_{t-1} + \sum_{i=f(t)+1}^{f(t+1)} w_0(i).$$

Proof. We re-express W_t by decomposing the weights:

$$\begin{aligned} W_t &= \sum_{i=1}^{f(t)} w_t(i) + \sum_{i=f(t)+1}^{f(t+1)} w_t(i) \\ &\leq \sum_{i=1}^{f(t)} w_t(i) + \sum_{i=f(t)+1}^{f(t+1)} w_0(i) && \text{(Definition of weight initialization)} \end{aligned}$$

Thus, it suffices to prove that $\sum_{i=1}^{f(t)} w_t(i) \leq W_{t-1} = \sum_{i=1}^{f(t)} w_{t-1}(i)$.

Let $I_t = \{1 \leq i \leq f(t) : w_t(i) \neq 0\}$. Using this set, we re-express our sum as

$$\begin{aligned}\sum_{i=1}^{f(t)} w_t(i) &= \sum_{i \in I_t} w_t(i) \\ &= 2 \sum_{i \in I_t, \hat{x}_t \notin L_i} w_{t-1}(i) + \sum_{i \in I_t, \hat{x}_t \in L_i} w_{t-1}(i) \qquad \text{(Definition of weight update)} \\ &= \sum_{i \in I_t} w_{t-1}(i) + \sum_{i \in I_t, \hat{x}_t \notin L_i} w_{t-1}(i)\end{aligned}$$

Let $I_{t-1} = \{i \leq f(t) : w_{t-1}(i) \neq 0\}$. Note that $I_t \subseteq I_{t-1}$. Because all weights are non-negative, we get an upper bound by replacing I_t with I_{t-1} in the above expression:

$$\begin{aligned}&\sum_{i \in I_t} w_{t-1}(i) + \sum_{i \in I_t, \hat{x}_t \notin L_i} w_{t-1}(i) \\ &\leq \sum_{i \in I_t} w_{t-1}(i) + \sum_{i \in I_{t-1}, \hat{x}_t \notin L_i} w_{t-1}(i) \\ &= \sum_{i \in I_{t-1}, x_t \in L_i} w_{t-1}(i) + \sum_{i \in I_{t-1}, \hat{x}_t \notin L_i} w_{t-1}(i) \\ &= W_{t-1} + \sum_{i \in I_{t-1}, x_t \in L_i} w_{t-1}(i) - \sum_{i \in I_{t-1}, \hat{x}_t \in L_i} w_{t-1}(i)\end{aligned}$$

By the definition of Algorithm 1, $\hat{x}_t$ is picked according to the rule

$$\hat{x}_t = \arg\max_{x \in \mathcal{X} \setminus x_{1:t-1}} \sum_{i=1}^{f(t)} w_{t-1}(i) \mathbf{1}[x \in L_i] = \arg\max_{x \in \mathcal{X} \setminus x_{1:t-1}} \sum_{i \in I_{t-1}, x \in L_i} w_{t-1}(i).$$

And by assumption, the adversary's output x_t comes from the set $\mathcal{X} \setminus x_{1:t-1}$. Thus, we are guaranteed that $\sum_{i \in I_{t-1}, x_t \in L_i} w_{t-1}(i) - \sum_{i \in I_{t-1}, \hat{x}_t \in L_i} w_{t-1}(i) \leq 0$ and can conclude that $\sum_{i=1}^{f(t)} w_t(i)$ is upper-bounded by W_{t-1}, completing the proof. □

We now apply this lemma to prove Theorem 4.1.

Proof of Theorem 4.1. We first note that unrolling Lemma A.1 gives a constant upper bound on any W_t:

$$W_t \leq W_{t-1} + \sum_{i=f(t)+1}^{f(t+1)} w_0(i) \leq \sum_{i=1}^{f(t+1)} w_0(i) \leq W.$$

For a particular true language L_i, the first step where L_i is considered in the argmax by the algorithm is at $f^{-1}(i) + 1$. Up until this point, the algorithm could have made at most $f^{-1}(i)$ mistakes.

We now focus on how many additional mistakes the algorithm can make after first introducing L_i. Suppose that at a timestep T, we have that

$$\sum_{t=f^{-1}(i)+1}^{T} \mathbf{1}[\hat{x}_t \notin L_i] = M,$$

i.e. the algorithm has made M mistakes after first introducing L_i. By definition of our update rule, for each of these timesteps, we had $x_t \in L_i$ because it is the true language, and $\hat{x}_t \notin L_i$. Thus, the initial weight of L_i has doubled M times, and we have

$$w_T(i) = 2^M w_0(i) \leq W_T \leq W.$$

Where we get an upper bound of W from Lemma A.1. Solving for M, we conclude that

$$M \leq \lfloor \log_2(W/w_0(i)) \rfloor.$$

Because this holds for any T, we conclude that the algorithm makes at most $\lfloor \log_2(W/w_0(i)) \rfloor$ additional mistakes after the introduction of L_i, and thus makes at most $f^{-1}(i) + \lfloor \log_2(W/w_0(i)) \rfloor$ mistakes in total. □

B Proofs from Section 5

B.1 Proof of Theorem 5.1

In this section, we prove that finite language classes can be generated with a number of mistakes logarithmic in the size of the class, and last-mistake-time bounded by the closure dimension. We prove this result by bringing together two algorithms with different guarantees. In particular, we consider the uniform generation algorithm proposed by Li et al. (2024) that guarantees a last-mistake-time upper-bounded by $\mathsf{Cdim}(\mathcal{L}) + 1$, but not necessarily any bound on the total number of mistakes, and the algorithm for finite language classes that we obtain via a reduction to Joshi et al. (2025), which upper-bounds the number of mistakes by $\log_2 |\mathcal{L}|$, but gives no guarantee on the last mistake time.

We show that our mistake-bounded algorithm can be viewed as a *sharpened* version of the uniform generation algorithm of Li et al. (2024), which coincides with the uniform generation algorithm at critical points, but makes smart decisions when there is leeway between which points to choose to still ensure the time-of-last-mistake guarantee.

We are thus able to combine the guarantees of both algorithms, and guarantee that the time-of-last-mistake is bounded by $\mathsf{Cdim}(\mathcal{L}) + 1$, while simultaneously guaranteeing that the total number of mistakes is at most $\log_2 |\mathcal{L}|$. Because the time-of-last-mistake is also an upper bound on the total number of mistakes, we can improve the total mistake bound to $\min\{\log_2 |\mathcal{L}|, \mathsf{Cdim}(\mathcal{L}) + 1\}$ when the closure dimension is small.

Algorithm 2 Uniform Generation Algorithm proposed by Li et al. (2024); Kleinberg & Mullainathan (2024)

Input: Universe $\mathcal{X}$, finite class $\mathcal{L}$
for $t = 1, \ldots, \infty$ **do**
  Compute $C = \{L \in \mathcal{L} : x_{1:t-1} \subseteq L\}$.
  If $|\bigcap_{L \in C} L \setminus x_{1:t-1}| > 0$, output $\hat{x}_t \in \bigcap_{L \in C} L \setminus x_{1:t-1}$. (1)
  Otherwise, output an arbitrary $\hat{x}_t \in \mathcal{X} \setminus x_{1:t-1}$.
end

Theorem B.1 (Uniform Generation Guarantee)**.** *(Kleinberg & Mullainathan, 2024; Li et al., 2024, ((2.2), Lemma 2.3, resp.)) For any finite class $\mathcal{L}$ and adversarial sequence of unique outputs consistent with $L \in \mathcal{L}$, the generator specified by Algorithm 2 has a last mistake time of at most* $\mathsf{Cdim}(\mathcal{L}) + 1$.

We highlight line (1) of Algorithm 2 as the key step necessary to obtain the last mistake time guarantee. Note that all other decisions made by the algorithm are completely arbitrary. This idea will be used to prove Theorem 5.1. In particular, we will give a mistake-bounded algorithm, and then show that it is consistent with the decisions of Algorithm 2 and thus also guarantees a last-mistake-time of $\mathsf{Cdim}(\mathcal{L}) + 1$.

Theorem 5.1. *For any finite class $\mathcal{L}$, there exists a generation algorithm that uniformly generates from $\mathcal{L}$ and guarantees that for any target $L^* \in \mathcal{L}$ and adversarial sequence consistent with L^*:*

1. *The total number of mistakes is at most* $\min\{\lfloor \log_2 |\mathcal{L}| \rfloor, \mathsf{Cdim}(\mathcal{L}) + 1\}$.
2. *The time of last mistake is at most* $\mathsf{Cdim}(\mathcal{L}) + 1$.

Proof of Theorem 5.1. Denote $\mathcal{L} = L_1, ..., L_k$, where $k = |\mathcal{L}|$. We instantiate Algorithm 1 with $w_0(i) = 1$ for all $i \in [k]$, and $f(t) = k$ for all $t \geq 1$. Thus, note that $f^{-1}(i) = 0$ for all $i \in [k]$.

By Theorem 4.1, under this instantiation, the algorithm is guaranteed to make at most $\lfloor \log_2 k \rfloor$ mistakes on any adversarial stream consistent with some $L \in \mathcal{L}$.

We now consider the time of last mistake bound. Due to the guarantees of Algorithm 2, it suffices to show that at any timestep t such that $C = \{L \in \mathcal{L} : x_{1:t-1} \subseteq L\}$ and $|\bigcap_{L \in C} L \setminus x_{1:t-1}| > 0$, the $\hat{x}_t$ output by our mistake-bounded algorithm, Algorithm 1, with the specified weight and growth functions, satisfies $\hat{x}_t \in \bigcap_{L \in C} L \setminus x_{1:t-1}$.

We note that by definition of our algorithm, for any $L_i \notin C$, we have $w_{t-1}(i) = 0$, and for all $L_j \in C$, $w_{t-1}(j) > 0$.

Thus, for any x in the set $\bigcap_{L \in C} L \setminus x_{1:t-1}$, the sum of the weights of its consistent languages achieves the maximum possible value, as it is the sum of all languages with non-zero weight, while any other x outside of the intersection must have strictly less total weight, because it is not included in one of the languages with non-zero weight.

We conclude that whenever $\left|\bigcap_{L \in C} L \setminus x_{1:t-1}\right| > 0$, the $\hat{x}_t$ output by Algorithm 1 defined as

$$\hat{x}_t = \arg\max_{x \in \mathcal{X} \setminus x_{1:t-1}} \sum_{i=1}^{k} w_{t-1}(i) \mathbf{1}[x \in L_i]$$

is guaranteed to satisfy $\hat{x}_t \in \bigcap_{L \in C} L \setminus x_{1:t-1}$.

This means that the outputs of Algorithm 1 are consistent with the decisions of Algorithm 2, and so we conclude that the algorithm not only provides a mistake bound, but also guarantees that the time of last mistake is at most $\mathsf{Cdim}(\mathcal{L}) + 1$. Because the number of mistakes can be at most the time of last mistake, this further strengthens the mistake bound to $\min\{\lfloor \log_2 k \rfloor, \mathsf{Cdim}(\mathcal{L}) + 1\}$. □

B.2 Proof of Lemma 5.2

We restate the lemma here for readability, and visualize the construction of a hard language class in Figure 1.

Lemma 5.2 (Lower Bound)**.** *For any $N \in \mathbb{N}$, there exists a finite language class $\mathcal{L}$ of size N such that no generation algorithm can guarantee fewer than $\lfloor \log_2 N \rfloor$ mistakes on every adversarial stream consistent with an $L \in \mathcal{L}$.*

Proof of Lemma 5.2. Our goal is to show that for any $n > 0$, there exists a language class $\mathcal{L}$ of size n such that every generator will make at least $\log_2 n$ mistakes against an adversarial stream from a language $L \in \mathcal{L}$.

Let $m \in \mathbb{Z}_{\geq 0}$ be the largest m such that $2^m \leq n$. Note that by definition, $m = \lfloor \log_2 n \rfloor$.

We construct a collection of 2^m languages $\mathcal{L}_m = \{L_v\}_{v \in \{0,1\}^m}$ where each language is identified with a bit string $v \in \{0,1\}^m$. If $2^m < n$, we additionally pad the language collection with dummy languages $L_{2^m+1}, ..., L_n$ that are disjoint from all the languages in $\mathcal{L}_m$ to ensure that the constructed collection has size exactly n. These will not be used in the construction of the hard adversarial stream.

We now define the contents of $\mathcal{L}_m$ as follows. Fix a subset of $\mathcal{X}$, B, of size $\sum_{i=1}^{m} 2^i = 2^{m+1} - 2$. We label each point with a bitstring of length between 1 and m: $B := \bigcup_{i=1}^{m} \{x_p : p \in \{0,1\}^i\}$.

A point $x_p \in B$ lies in $L_v \in \mathcal{L}_m$ if and only if p is a prefix of v.

In order to ensure each language has infinite support, we pad each L_v with an infinite subset of $\mathcal{X}$, disjoint from all other languages.

We now describe how an adversary can construct an adversarial stream that forces at least m mistakes by the generator.

The adversary keeps track of a bitstring $s = \{\}$, which initially begins as empty.

The adversary loops the following process for each time $t = 1, ...$:

1. Observe the generator's output $\hat{x}_t$.

2. If $t = m + 1$, BREAK from the loop, and enumerate all remaining elements of L_s.

3. If the generator outputs $x_v \in B$ or $x \in L_v \setminus B$ such that s is a prefix of v, observe the bit immediately proceeding s: $v[|s|]$ (0-indexed), and append the *complement* of that bit to s. Otherwise, append an arbitrary bit to s.

4. Output $x_s \in B$.

It remains to show that the adversary's stream is valid, i.e. it consists of all unique elements and is consistent with a language in $\mathcal{L}_m$, and that the generator makes m mistakes.

The adversary's stream is valid First, we consider the final bitstring s of length m that the adversary constructs before hitting the break at time $m + 1$. At this point, $x_s \in B$ is consistent with exactly one language, L_v where $v = s$, as it is the only string of length m that contains s as a prefix. Moreover, we are guaranteed that all x_t for $t \leq m$ satisfied $x_t \in L_v$, because the bitstring s at that time was a prefix of the final s by construction, and all were unique, because s was a different length at each step. By definition, all of the adversary's outputs after time $t = m$ are also unique and consistent. Therefore, we conclude that the adversarial stream is valid, and moreover is actually a valid *enumeration* of L_v.

The generator makes m mistakes. We can show that every $\hat{x}_t$ for $t \leq m$ is a mistake. In particular, at the beginning of timestep t, the set of potential languages that are consistent with the adversarial stream so far are exactly all of the languages L_v where s is a prefix of v. By construction, there are 2^{m-t+1} such languages. The adversary's x_t is defined to be $x_{s||0}$ or $x_{s||1}$, where seeing either of these points reduces the set of consistent languages by half to 2^{m-t}, and moreover these two consistent sets are totally disjoint.

If in the check in step 3, the generator outputs $x_v \in B$ or $x \in L_v \setminus B$ such that s is a prefix of v, this means that its output element can only be consistent with languages in one of these two disjoint sets. By playing $x_{s||\neg v[|s|]}$, the adversary reveals the true language to be in the other set of consistent languages, and thus the generator has made a mistake. If the generator's point does not satisfy the first condition, it means the generated element is not consistent with any of the 2^{m-t} languages still consistent with the adversary's stream, and thus is also automatically a mistake regardless of what the adversary outputs.

Thus, we conclude that in every step from $t = 1$ to $t = m$, the generator's move does not lie in the target language L_v, and thus the generator makes $m = \lfloor \log_2 n \rfloor$ mistakes. □

C Proofs from Section 6

C.1 Proof of Theorem 6.1

Theorem 6.1. *Let $\mathcal{L} = \{L_1, L_2, ...\}$ be a countably infinite class of languages under a fixed enumeration. For any language $L_i \in \mathcal{L}$, the instantiation of Algorithm 1 described above guarantees a non-uniform mistake bound of:*

$$M_{\text{non-unif}}(\mathcal{G}, i) \leq 3 \log_2 i + \log_2(\pi^2/6) = O(\log_2 i).$$

Proof of Theorem 6.1. To prove the theorem, we instantiate Theorem 4.1 with the weight function $w_0(i) = 1/i^2$ and growth function $f(t) = 2^t$. By standard results due to Euler (1740), we have that $\sum_{i=1}^{\infty} 1/i^2 = \pi^2/6$, and thus can set $W = \pi^2/6$.

Under this definition of f, $f^{-1}(i)$, i.e. the largest t such that $f(t) < i$, is upper bounded by $\log_2(i)$.

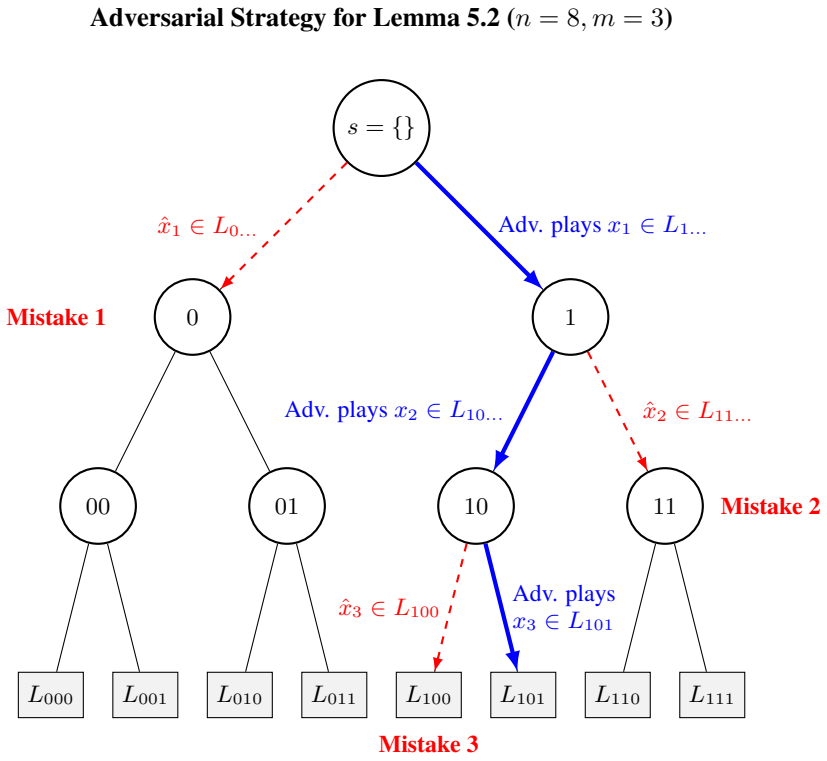


Figure 1: Visualizing the proof of Lemma 5.2. The tree represents the hierarchy of languages in $\mathcal{L}$. At each internal node s, the generator is forced to output a prediction consistent with exactly one branch, corresponding to halving the number of consistent languages (red dashed arrow). The adversary observes this and outputs an element that forces the true language to lie in the *opposite* branch (blue solid arrow), guaranteeing a mistake. This process repeats for $m = \lfloor \log_2 |\mathcal{L}| \rfloor$ steps.

Thus, for this particular instantiation of weight functions and growth function, we can guarantee that Algorithm 1 guarantees a non-uniform mistake bound of

$$\begin{aligned} M_{\text{non-unif}}(\mathcal{G}, i) &\leq f^{-1}(i) + \lfloor \log_2(W/w_0(i)) \rfloor \\ &\leq \lfloor \log_2 i \rfloor + \log_2(\pi^2 i^2/6) \\ &\leq \log_2(\pi^2/6) + 3 \log_2 i \\ &= O(\log_2 i). \end{aligned}$$

□

D Proof of Lemma 6.2

In this section, we provide the details of the proof of Lemma 6.2, which guarantees the existence of an algorithm that matches the last-mistake-time guarantees of the non-uniform generation algorithm of Charikar & Pabbaraju (2024), while guaranteeing a total number of mistakes that is linear in the index of the target language:

Lemma 6.2. *There exists a non-uniform generation algorithm (*Modified-Greedy*) with a time-of-last-mistake guarantee of* $\max\{i-1, m(L_i)+1\}$ *and a non-uniform mistake bound of:*

$$M_{\text{non-unif}}(\mathcal{G}, i) \leq \min\{2(i-1), \max\{i-1, m(L_i)+1\}\}.$$

We refer to the algorithm we will use to obtain our guarantees as Modified-Greedy (Algorithm 3). The algorithm as we describe it differs from the original definition provided by Charikar & Pabbaraju (2024), with the changed lines marked with "$^{\triangle}$". However, while the modification means that

the actions taken by the two algorithms will be different, the modified algorithm still satisfies the last-mistake-bound promised by Charikar & Pabbaraju (2024). For completeness, we re-prove the last-mistake-time bound for the modified algorithm in Lemma D.1.

Algorithm 3 MODIFIED-GREEDY (c.f. Algorithm in Theorem 6 of Charikar & Pabbaraju (2024)) Modified lines are marked with "$^{\triangle}$."

Input: Universe $\mathcal{X}$, infinite stream $\mathcal{L} = L_1, L_2, ...$
for $t = 1, \ldots, \infty$ **do**
  Initialize $I_t = \mathcal{X} \setminus x_{1:t-1}$.$^{\triangle}$
  for $i = 1, ..., t$ **do**
    if $x_{1:t-1} \subseteq L_i$ *and* $|L_i \cap I_t| > 0$ $^{\triangle}$ **then**
      $I_t \leftarrow I_t \cap L_i$
    end
  end
  Output $\hat{x}_t \in I_t$.
end

Lemma D.1. *For any infinite stream $\mathcal{L} = L_1, L_2, ...,$* MODIFIED-GREEDY*(Algorithm 3) guarantees a last mistake time $t(L_i) \leq \max\{i-1, m(L_i)+1\}$ for any adversarial stream consistent with $L_i \in \mathcal{L}$.*

Proof. Consider any $t \geq \max\{i, m(L_i)+2\}$. This means that during step t, MODIFIED-GREEDY considers L_i as one of the languages to possibly intersect with I_t. We are also guaranteed that $x_{1:t-1} \subseteq L_i$ because the adversary is consistent with L_i.

Consider the point at which L_i is considered by the algorithm during step t. By definition, there is some $J \subseteq [i-1]$ such that $x_{1:t-1} \subseteq L_j$ for each $j \in J$, and $I_t = \bigcap_{j \in J} L_j \setminus x_{1:t-1}$. Thus, $x_{1:t-1} \subseteq \bigcap_{j \in J} L_j \cap L_i$.

If $|I_t \cap L_i| > 0$, the algorithm is guaranteed to play a $\hat{x}_t \in L_i$, and thus does not make a mistake. We claim that it cannot be the case that $|I_t \cap L_i| = 0$. This is because this would imply that

$$t - 1 \leq \left| \bigcap_{j \in J} L_j \cap L_i \right| \leq m(L_i),$$

but this is a contradiction, because we assumed that $t \geq m(L_i) + 2$.

Thus, we conclude that MODIFIED-GREEDY will not make a mistake for all

$$t \geq \max\{i, m(L_i)+2\},$$

and thus the last mistake time is at most $\max\{i-1, m(L_i)+1\}$. □

We are now ready to prove Lemma 6.2, which further bounds the number of mistakes made by MODIFIED-GREEDY to show that it satisfies both a time-of-last-mistake bound and a total mistake bound.

Proof of Lemma 6.2. Our proof proceeds by showing that every mistake incurred by MODIFIED-GREEDY on a language L_i is accompanied by eliminating some language in the prefix $L_1, ..., L_{i-1}$. Because there are only $i-1$ such languages, the number of mistakes on L_i is bounded by $2(i-1)$, where the double factor comes because we don't consider L_i until step i, and so could make mistakes during the $i-1$ steps before that.

At any step $t \geq i$, we have two cases. Either $\hat{x}_t \in L_i$, in which we can't make a mistake, or $\hat{x}_t \notin L_i$, in which case the algorithm makes a mistake when the true language is L_i.

if $\hat{x}_t \notin L_i$, this means that at the point in the algorithm during step t when $I_t \cap L_i$ was considered, we had $|I_t \cap L_i| = 0$. Note that by definition, $I_t = \bigcap_{j \in J} L_j \setminus x_{1:t-1}$ for some $J \subseteq [i-1]$ satisfying $x_{1:t-1} \subseteq L_j$ for each $j \in J$.

Thus, because $x_t \in L_i$, we have $x_t \notin I_t$, which implies that there exists some $j < i$ such that $x_{1:t-1} \subseteq L_j$, but $x_t \notin L_j$. This implies that L_j becomes inconsistent at step t, and is no longer considered after t.

Because there are only $i-1$ languages before i, this means that the algorithm can make at most $i-1$ mistakes after starting to consider L_i, and thus can make at most $2(i-1)$ mistakes.

Finally, the last-mistake-time of $\max\{i-1, m(L_i)+1\}$ follows from Lemma D.1, and because the total number of mistakes is upper-bounded by the last-mistake time, we can refine the total number of mistakes to $\min\{2(i-1), \max\{i-1, m(L_i)+1\}\}$. □

D.1 Proof of Theorem 6.3

Theorem 6.3. *There exists an infinite language class with a fixed enumeration $\mathcal{L} = \{L_1, L_2, \ldots\}$ such that, for every generation algorithm $\mathcal{G}$ satisfying*

$$t_{\mathcal{G}}(L_i) \leq C_{\mathcal{G}} \max\{i, m(L_i)\}$$

for some finite constant $C_{\mathcal{G}}$ and every i, there exists a constant $K_{\mathcal{G}}$ such that

$$M_{\text{non-unif}}(\mathcal{G}, i) \geq i - K_{\mathcal{G}}$$

for every $i > K_{\mathcal{G}}$. In particular,

$$M_{\text{non-unif}}(\mathcal{G}, i) = \Omega(i).$$

Proof of Theorem 6.3. We construct our hard language stream on the countably infinite $\mathcal{X} = \mathbb{N} \times \mathbb{N}$, and thus refer to elements $x \in \mathcal{X}$ as pairs $(i, j) \in \mathbb{N} \times \mathbb{N}$.

We define language L_i as follows for any $i \geq 1$:

$$L_i = \{(i, j) : j \in \mathbb{N}\} \cup \bigcup_{k \in [i-1]} \{(k, j) : j \in [2^{2^k}]\}$$

Note that this is a valid language class, because each language has infinite support, and for any particular language L_i,

$$m(L_i) = |\bigcup_{k \in [i-1]} \{(k, j) : j \in [2^{2^k}]\}| = \sum_{j=1}^{i-1} 2^{2^j}$$

With $m(L_1) = 0$. Similarly note that

$$|\bigcap_{j \geq i} L_j| = m(L_{i+1}) = m(L_i) + 2^{2^i} = \omega(m(L_i)).$$

We have the adversary first enumerate all $x \in \bigcap_{i \geq 1} L_i = \{(1, j) : j \in [2^{2^1}]\}$, then all $x \in \bigcap_{i \geq 2} L_i \setminus L_1 = \{(2, j) : j \in [2^{2^2}]\}$, then all $x \in \bigcap_{i \geq 3} L_i \setminus L_2 = \{(3, j) : j \in [2^{2^3}]\}$ etc.

Let $t_i = m(L_i) + 1$ be the timestep at which the adversary would output $x_t = (i, 1)$ for each $i \in \mathbb{N}$. We call this a *critical boundary*, as these are the only timesteps at which the set of consistent languages shrinks, and in particular, ceases to contain L_{i-1}. In particular, at every t_i, before seeing the adversary's output, the generator is forced choose between an x that is consistent with L_{i-1} but inconsistent with all languages that remain consistent (L_j for $j \geq i$), or output a point that is consistent with L_j for all $j \geq i$, but inconsistent with L_{i-1}.

By the guarantee of our generator, we have $t_{\mathcal{G}}(L_{i-1}) \leq C_{\mathcal{G}} \max\{i-1, m(L_{i-1})\}$, and thus because $m(L_i) = \omega(m(L_{i-1}))$, there must be some finite index $i_0(\mathcal{G})$ such that for all $j \geq i_0(\mathcal{G})$, $t_j > C_{\mathcal{G}} \max\{j-1, m(L_{j-1})\} \geq t_{\mathcal{G}}(L_{j-1})$.

Consider any $i^* \geq i_0(\mathcal{G})$, and apply the following adversarial strategy for the remainder of the enumeration for all $j \geq i_0(\mathcal{G})$:

1. if $\hat{x}_{t_j} \notin L_{j-1}$, the adversary stops its enumeration, and instead enumerates all of the remaining elements of L_{j-1}.
2. Otherwise, if $j = i^*$, the adversary stops and enumerates all remaining elements of L_{i^*}.

Note that we are guaranteed to trigger one of steps 1 or 2 at some point during the game, as i^* is finite. If Step 1 executes, this implies a contradiction to the last-mistake-time bound for language L_{j-1}, as all of the adversary's outputs thus far and in the future are consistent with L_{j-1}, but the generator makes a mistake on L_{j-1} at time

$$t_j > t_{\mathcal{G}}(L_{j-1}).$$

Note that by construction, at timestep t_j, we have that for any $j' \geq j$, $L_{j'} \cap L_{j-1} \setminus x_{1:t_j-1} = \emptyset$, and thus if the generator plays $\hat{x}_{t_j} \in L_{j-1}$, we must have $\hat{x}_{t_j} \notin L_{j'}$ for all $j' \geq j$, and thus this counts as a mistake if any such j' ends up being the target language.

If step 1 never executes, this means that the generator plays from L_{j-1} at each t_j for $j = i_0(\mathcal{G}), ..., i^*$, and thus makes at least $i^* - i_0(\mathcal{G}) + 1$ mistakes on L_{i^*}.

Thus, in either case the generator is forced to either contradict the last-mistake-time-bound of $m(L_i)$ for some $i \leq i^*$, or make a linear number of mistakes on i^*.

Thus, we conclude that taking $K_{\mathcal{G}} := i_0(\mathcal{G}) - 1$, if $\mathcal{G}$ satisfies $t_{\mathcal{G}}(L_i) \leq C_{\mathcal{G}} \max\{i, m(L_i)\}$ for all i, then indeed for any $i > K_{\mathcal{G}}$,

$$M_{\text{non-unif}}(\mathcal{G}, i) \geq i - K_{\mathcal{G}} = \Omega(i).$$

□

E Mistake Bounds via Learning from Demonstrations

Recent work by Joshi et al. (2025) studies the setting of Learning from Demonstrations (LfD). In this section, we discuss how the problem of mistake bounded language generation can be reduced to this setting, allowing us to use the techniques of Joshi et al. (2025) to derive mistake bounds.

The online setting of learning from correct demonstrations can be thought of as a contextual bandit problem. Let $\mathcal{X}$ be a set of contexts, and $\mathcal{Y}$ be a set of actions. A reward function $r : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ specifies a binary reward for each context/action pair.

The online interaction between the learner and adversary proceeds as follows for each round $t = 1, 2, 3, ..., T$:

1. The learner receives context $x_t \in \mathcal{X}$ from the adversary.
2. The learner outputs an action $\hat{y}_t \in \mathcal{Y}$.
3. The learner receives a demonstration $y_t \in \mathcal{Y}$ from the adversary. *The learner receives no other feedback.*

In the setting of learning from an optimal demonstrator, or what can be thought of as the "realizable" setting of the problem, the learner has access to a collection of reward functions $\mathcal{R}$. The adversary is assumed to be optimal, in that there exists some true reward function $r^* \in \mathcal{R}$ such that for all timesteps t, $r^*(x_t, y_t) = 1$. We assume that all reward functions and contexts have at least one action that results in non-zero reward.

Crucially, this true reward r^* is unknown to the learner and the learner never receives feedback about the reward values of its output actions. Nevertheless, the learner's performance is evaluated with respect to this true reward. We say that the learner makes a *mistake* at timestep t if $r^*(x_t, \hat{y}_t) = 0$.

Typical online learning strategies typically rely on feedback at every timestep in order to update a policy toward more successful outputs. The lack of such feedback in this online demonstration setup thus makes it seem difficult, and surprising that a learner could hope to bound its mistakes. However, this is the main result of Joshi et al. (2025), who show that for finite reward classes $\mathcal{R}$, there exists an algorithm that guarantees a learner makes no more than $\log_2 |\mathcal{R}|$ mistakes on any $\mathcal{R}$:

Theorem E.1 (Joshi et al. (2025), Theorem 4)**.** *There exists an online algorithm such that given any online sequence $((x_t, y_t))_{t\in\mathbb{N}}$ such that there exists $r^* \in \mathcal{R}$ with $r^*(x_t, y_t) = 1$ for all t, the algorithm makes at most $\log_2 |\mathcal{R}|$ mistakes, e.g.*

$$\sum_{t\in\mathbb{N}}(1 - r^*(x_t, \hat{y}_t)) \leq \log_2 |\mathcal{R}|.$$

E.1 Connecting Mistake-Bounded Language Generation to Learning from Demonstrations

In this section, we describe how guarantees for learning from demonstrations can be translated to mistake bounds for language generation. At first glance, one might assume that language generation is a special case of learning from demonstrations where there is only a single context, and each reward function is an indicator of a particular language.

However, language generation's requirement that the generator output *unseen* elements means that this is not quite the case, as a learner in the single context setting could succeed by continuously outputting the adversary's initial output. To draw a true comparison, we must carefully construct contexts and reward functions based on a language class to ensure a learner can only receive high reward by outputting new elements. We describe the full reduction in the theorem below.

We present the reduction in the most general terms possible, i.e. for potentially non-uniform mistake bounds dependent on the position of the reward function in the stream and an imperfect adversary, as well as in terms of the timestep T. For the remainder of this section, we denote the universe of elements used in language generation as $\mathcal{U}$, with elements $u \in \mathcal{U}$, to differentiate from the context space in LfD.

Lemma E.2. *Suppose we have access to a learning from demonstrations algorithm such that for any choice of contexts $\mathcal{X}$, label set $\mathcal{Y}$, and reward class $\mathcal{R} = r_1, r_2, ...$, presented as a (potentially infinite) stream of functions, the algorithm guarantees that for any adversarial stream $(x_1, y_1), (x_2, y_2), ...$, the following mistake bound is satisfied for each $r_i \in \mathcal{R}$ and $T \in \mathbb{N}$:*

$$\sum_{t=1}^{T}(1 - r_i(x_t, \hat{y}_t)) \leq M(i, T, \sum_{t=1}^{T}(1 - r_i(x_t, y_t))).$$

Then, for any language class $\mathcal{L} = L_1, L_2, ...$, we can use this algorithm to construct a generator whose outputs are guaranteed to satisfy for any $L_i \in \mathcal{L}$ and $T \in \mathbb{N}$,

$$\sum_{t=1}^{T}\mathbf{1}[\hat{u}_t \notin L_i \setminus u_{1:t-1}] \leq M(i, T, \sum_{t=1}^{T}\mathbf{1}[u_t \notin L_i \setminus u_{1:t-1}]).$$

Joshi et al. (2025)'s result for finite classes and perfect demonstrators can be interpreted in this setting as guaranteeing $M(i, T, 0) \leq \log_2 |\mathcal{R}|$ for all $i \in [|\mathcal{R}|]$ and $T \in \mathbb{N}$, and thus bounds the number of mistakes in the language setting by $M(i, T, 0) \leq \log_2 |\mathcal{L}|$ as well.

In the setting of sub-optimal demonstrators, where the demonstrator may not always give a demonstration with high reward, Joshi et al. (2025) show that when the reward class is finite, the sub-optimality of the learner's reward with respect to r_i can be upper-bounded in terms of the class size and adversary's suboptimality, thus guaranteeing

$$M(i, T, \sum_{t=1}^{T}(1 - r_i(x_t, y_t))) \leq (1 + 2\gamma)\sum_{t=1}^{T}(1 - r_i(x_t, y_t)) + \log_2 |\mathcal{R}|/\gamma$$

for any choice of $\gamma \in (0, 3/4]$. This is the guarantee we leverage to provide mistake-bound guarantees for noisy adversaries in Section 7.

The results of Joshi et al. (2025) only apply to settings with finite reward classes, where we can hope for a uniform guarantee over all possible reward functions. However, the setting of non-uniform language generation requires mistake bounds that can be guaranteed non-uniformly over a countably infinite stream of languages.

While our reduction still applies to this case, there are no existing results that provide LfD guarantees for an infinite stream of rewards. In Section E.3, we give a new algorithm for the LfD setting that can give guarantees for an infinite stream of reward functions, allowing us to derive mistake bounds in the non-uniform language generation setting.

We now present the proof of Lemma E.2.

Proof of Lemma E.2. Given a language class $\mathcal{L} = L_1, L_2, ...$ for a universe $\mathcal{U}$ and a general algorithm for LfD, we describe how to leverage the LfD algorithm to generate with bounded mistakes.

We first define the context space $\mathcal{X}$ and label space $\mathcal{Y}$ for the algorithm. We let $\mathcal{Y} = \mathcal{U}$, and $\mathcal{X} = \{x_{u_{1:n}} : n \geq 0, u_1, ..., u_n \in \mathcal{U}\}$, e.g. a unique context for each finite string of elements from $\mathcal{U}$. We refer to elements in $\mathcal{X}$ as x_s, where s is some finite string of elements $\{u_1, ..., u_n\}$.

For a language L_i, we define a corresponding reward function r_i as follows:

$$r_i(x_s, y) = \mathbf{1}[y \in L_i \setminus s].$$

Our reward function class of interest is thus $\mathcal{R} = r_1, r_2, ...$, where each r_i is identified with $L_i \in \mathcal{L}$.

We now describe how to use a learner for the above problem to generate with bounded mistakes. At the beginning of each step t, the elements that have been shown thus far by the adversary are some $u_1, ..., u_{t-1}$.

To leverage the LfD algorithm, we act as the adversary and show the learner the context $x_{u_{1:t-1}}$. The learner outputs some $\hat{y}_t \in \mathcal{U}$. If $\hat{y}_t$ is unseen, we generate exactly this element in the language generation setting. Otherwise, we replace $\hat{y}_t$ with an arbitrary unseen element. We then see the adversary-generated element u_t, and make this the adversarial demonstration $y_t = u_t$ in the LfD algorithm.

Note that by definition, the noise of our language generation adversary on a particular language coincides with the sub-optimality of the demonstrator on the corresponding reward function, i.e. for any $L_i \in \mathcal{L}$ and $T \in \mathbb{N}$,

$$\sum_{t=1}^{T} \mathbf{1}[u_t \notin L_i \setminus u_{1:t-1}] = \sum_{t=1}^{T} (1 - r_i(x_{u_{1:t-1}}, y_t)).$$

Similarly, the mistakes of our generator on L_i are upper-bounded by the sub-optimality of the learner on r_i:

$$\sum_{t=1}^{T} \mathbf{1}[\hat{u}_t \notin L_i \setminus u_{1:t-1}] \leq \sum_{t=1}^{T} (1 - r_i(x_{u_{1:t-1}}, \hat{y}_t)).$$

The above inequality holds with equality if the learner only outputs unseen elements and we never need to replace $\hat{y}_t$ with an arbitrary unseen element.

Thus, we conclude that if the LfD algorithm guarantees that at a particular step T that for any i, the learner's actions satisfy the guarantee

$$\sum_{t=1}^{T} (1 - r_i(x_t, \hat{y}_t)) \leq M(i, T, \sum_{t=1}^{T} (1 - r_i(x_t, y_t))),$$

then the outputs of our constructed generator must also satisfy

$$\sum_{t=1}^{T} \mathbf{1}[\hat{u}_t \notin L_i \setminus u_{1:t-1}] \leq M(i, T, \sum_{t=1}^{T} \mathbf{1}[u_t \notin L_i \setminus u_{1:t-1}]).$$

□

E.2 Learning from Sub-Optimal Demonstrators and Noisy Adversaries

In this section, we describe how to derive our mistake bounds for noisy adversaries from Lemmas 7.1 and 7.2. Both results can be derived by leveraging the reduction outlined in Lemma E.2 along with an appropriate result from the LfD setting. The finite case immediately follows from an existing result of Joshi et al. (2025), while Lemma 7.2 follows from proper instantiation of Theorem E.5.

E.2.1 Deriving Lemma 7.1

Joshi et al. (2025) give the following result for learning from potentially sub-optimal demonstrations for a finite class of reward functions.

Theorem E.3 (Theorem 5, Joshi et al. (2025))**.** *There exists an algorithm such that for any finite $\mathcal{R}$ such that $\max_y r(x, y) = 1$ for all $r \in \mathcal{R}$ and $x \in \mathcal{X}$, and any online sequence (x_t, y_t), after the end of T rounds, for any $0 < \gamma \leq 3/4$ and for any $r \in \mathcal{R}$,*

$$\sum_{t=1}^{T}(1 - r(x_t, \hat{y}_t)) \leq (1 + 2\gamma)\sum_{t=1}^{T}(1 - r(x_t, y_t)) + \log_2 |\mathcal{R}|/\gamma.$$

Combining this result with the reduction in Lemma E.2 immediately gives the mistake bound for noisy adversaries stated in Lemma 7.1.

E.2.2 Proof of Lemma 7.2

Lemma 7.2 (Noisy Mistake Bound for Infinite Streams)**.** *Suppose we are given an infinite stream of languages $\mathcal{L} = L_1, L_2, ...$ and $\gamma \in (0, 3/4]$. Then, there exists a generation algorithm such that for every adversarial stream $x_1, x_2, ...$, not necessarily consistent with any $L_i \in \mathcal{L}$ and $i, T \in \mathbb{N}$,*

$$\sum_{t=1}^{T}\mathbf{1}[\hat{x}_t \notin L_i] \leq (1 + 2\gamma)\sum_{t=\max\{\lfloor \log_2 i \rfloor, 1\}}^{T}\mathbf{1}[x_t \notin L_i] + (1 + 2/\gamma)\log_2 i + \frac{\log_2(\pi^2/6)}{\gamma}.$$

Proof of Lemma 7.2. We instantiate Algorithm 4 with polynomially decreasing weights $w_0(i) = 1/i^2$ and exponential growth function $f(t) = 2^t$. Note that this implies

$$f^{-1}(i) = \max\{0, \lceil \log_2 i \rceil - 1\} \leq \log_2 i.$$

Moreover,

$$\max\{1, \lfloor \log_2 i \rfloor\} \leq f^{-1}(i) + 1.$$

Thus, because the demonstrator-regret terms are nonnegative, we may replace the lower endpoint $f^{-1}(i) + 1$ by $\max\{1, \lfloor \log_2 i \rfloor\}$ to obtain a looser upper bound.

Plugging this in to the guarantee of Theorem E.5, we derive the following guarantee for general binary rewards. Note that because all languages have infinite support, we can assume that $\sup_{y \in \mathcal{Y}} r_i(x, y) = 1$ for all $i \in \mathbb{N}$ and $x \in \mathcal{X}$. Using the same reasoning as in Theorem 6.1, this simplifies to

$$\sum_{t=1}^{T}(1 - r_i(x_t, \hat{y}_t)) \leq \frac{\log_2(\pi^2/6)}{\gamma} + (1 + 2\gamma)\sum_{t=\max\{\lfloor \log_2 i \rfloor, 1\}}^{T}(1 - r_i(x_t, y_t)) + (1 + 2/\gamma)\log_2 i$$

Substituting this into our reduction (Lemma E.2), we conclude that we can construct a generator with the following guarantee for every T and i:

$$\sum_{t=1}^{T}\mathbf{1}[\hat{x}_t \notin L_i] \leq (1 + 2\gamma)\sum_{t=\max\{\lfloor \log_2 i \rfloor, 1\}}^{T}\mathbf{1}[x_t \notin L_i] + (1 + 2/\gamma)\log_2 i + \frac{\log_2(\pi^2/6)}{\gamma},$$

Thus matching the promised guarantee.

□

E.3 Learning from Demonstrations with Infinite Reward Functions

While we can directly leverage the results of Joshi et al. (2025) to derive mistake bounds for finite language classes via our reduction in Lemma E.2, the problem of language generation is particularly concerned with settings in which we must learn to generate well from a *countably infinite stream* of languages. In this section, we extend the results of Joshi et al. (2025) and give an algorithm that provides non-uniform guarantees for an infinite stream of reward functions, $\mathcal{R} = r_1, r_2, \ldots$.

Algorithm 4 Learning from Arbitrary Demonstrators with an Infinite Stream of Reward Functions.

Input: Context space $\mathcal{X}$, label space $\mathcal{Y}$, infinite stream of reward functions $\mathcal{R} = \{r_1, r_2, \dots\}$ (with finite-range $r : \mathcal{X} \times \mathcal{Y} \to [0,1]$, assumed binary when $\gamma = 1$), weights $w_0 : \mathbb{N} \to \mathbb{R}_{\geq 0}$, growth function $f : \mathbb{N} \to \mathbb{N}$, parameter $\gamma \in (0, 3/4] \cup \{1\}$

for $t = 1, 2, \ldots, \infty$ **do**

Observe x_t

Output $\hat{y}_t := \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{f(t)} w_{t-1}(i) r_i(x_t, y)$

Observe adversary's y_t

`// Update weights for` $i \in [f(t)]$

if $\gamma \in (0, 3/4]$ **then**

$w_t(i) \leftarrow w_{t-1}(i)(1+\gamma)^{\lambda_i(x_t, \hat{y}_t)}(1-\gamma)^{\lambda_i(x_t, y_t)}$

where $\lambda_i(x, y) := \sup_{y' \in \mathcal{Y}} r_i(x, y') - r_i(x, y)$

else if $\gamma = 1$ **then**

$$w_t(i) \leftarrow \begin{cases} 0 & \text{if } r_i(x_t, y_t) = 0 \\ w_{t-1}(i) & \text{if } r_i(x_t, y_t) = r_i(x_t, \hat{y}_t) = 1 \\ 2w_{t-1}(i) & \text{if } r_i(x_t, y_t) = 1, r_i(x_t, \hat{y}_t) = 0 \end{cases}$$

end

`// Initialize weights for` $i \in \{f(t)+1, \ldots, f(t+1)\}$

if $\gamma \in (0, 3/4]$ **then**

$w_t(i) = w_0(i)$

else if $\gamma = 1$ **then**

$$w_t(i) = \begin{cases} w_0(i) & \text{if } r_i(x_j, y_j) = 1 \text{ for all } j \in [t] \\ 0 & \text{otherwise} \end{cases}$$

end

end

It will be helpful to define the following notation. For any $i \in \mathbb{N}$ and $T \in \mathbb{N}$, denote

$$\hat{M}^T_{i,alg} := \sum_{t=1}^{T} r_i(x_t, \hat{y}_t), \quad M^T_{i,opt} := \sum_{t=1}^{T} \sup_{y \in \mathcal{Y}} r_i(x_t, y),$$

$$R_i(T' : T) = \sum_{t=T'}^{T} \left(\sup_{y \in \mathcal{Y}} r_i(x_t, y) - r_i(x_t, y_t) \right).$$

Note that in the binary rewards case, if we assume that every context has at least one high-reward label, $M^T_{i,opt}$ is always equal to T.

We present the following two results about Algorithm 4, one for the optimal demonstrator and binary reward case, and the other for sub-optimal demonstrators.

Theorem E.4. *For any infinite stream $\mathcal{R} = r_1, r_2, \ldots$ of binary reward functions, and any online sequence (x_t, y_t) satisfying $r_i(x_t, y_t) = 1$ for some $i \in \mathbb{N}$ and all $t \in \mathbb{N}$, initial weight function $w_0 : \mathbb{N} \to \mathbb{R}_{>0}$ satisfying $\sum_{i=1}^{\infty} w_0(i) \leq W < \infty$, non-decreasing growth function $f : \mathbb{N} \to \mathbb{N}$, after the end of T rounds, Algorithm 4 with input $\gamma = 1$ guarantees that for any r_i such that $r_i(x_t, y_t) = 1$*

for all $t \in [T]$,

$$\sum_{t=1}^{T}(1 - r_i(x_t, \hat{y}_t)) \leq \log_2(W/w_0(i)) + f^{-1}(i).$$

Theorem E.5. *For any infinite stream* $\mathcal{R} = r_1, r_2, ...$ *where each* r_i *has finite range, any online sequence* (x_t, y_t)*, initial weight function* $w_0 : \mathbb{N} \to \mathbb{R}_{>0}$ *satisfying* $\sum_{i=1}^{\infty} w_0(i) \leq W < \infty$*, non-decreasing growth function* $f : \mathbb{N} \to \mathbb{N}$*, and* $\gamma \in (0, 3/4]$*, and any* $i \in \mathbb{N}$*, after the end of* T *rounds, Algorithm 4 guarantees that*

$$M_{i,opt}^{T} - \hat{M}_{i,alg}^{T} \leq \log_2(W/w_0(i))/\gamma + (1 + 2\gamma)R_i(f^{-1}(i) + 1 : T) + f^{-1}(i).$$

The proofs of Theorems E.4 and E.5 use a similar potential argument to that of Theorem 4.1. We begin by bounding the growth of the total weight using the following lemma:

Lemma E.6. *Define* $W_t := \sum_{i=1}^{f(t+1)} w_t(i)$*. Then, for any step* $t \geq 1$*,*

$$W_t \leq W_{t-1} + \sum_{i=f(t)+1}^{f(t+1)} w_0(i).$$

Proof. We re-express W_t by decomposing the weights:

$$\begin{aligned} W_t &= \sum_{i=1}^{f(t)} w_t(i) + \sum_{i=f(t)+1}^{f(t+1)} w_t(i) \\ &\leq \sum_{i=1}^{f(t)} w_t(i) + \sum_{i=f(t)+1}^{f(t+1)} w_0(i) && \text{(Definition of weight initialization)} \end{aligned}$$

Thus, it suffices to prove that $\sum_{i=1}^{f(t)} w_t(i) \leq W_{t-1} = \sum_{i=1}^{f(t)} w_{t-1}(i)$.

When $\gamma = 1$ and the rewards are binary, this inequality follows directly from the argument of Lemma A.1, with the round-t set

$$L_i^{(t)} := \{y \in \mathcal{Y} : r_i(x_t, y) = 1\}$$

playing the role of L_i. We therefore assume throughout the remainder of the proof that $\gamma \in (0, 3/4]$.

We go about this by showing that $\sum_{i=1}^{f(t)} w_t(i) - W_{t-1} \leq 0$:

$$\begin{aligned} &\sum_{i=1}^{f(t)} (w_t(i) - w_{t-1}(i)) \\ &= \sum_{i=1}^{f(t)} \Big(w_{t-1}(i)(1+\gamma)^{\lambda_i(x_t, \hat{y}_t)}(1-\gamma)^{\lambda_i(x_t, y_t)} - w_{t-1}(i) \Big) \\ &\leq \sum_{i=1}^{f(t)} (w_{t-1}(i)(1 + \gamma\lambda_i(x_t, \hat{y}_t))(1 - \gamma\lambda_i(x_t, y_t)) - w_{t-1}(i)) \end{aligned}$$

where the upper bound follows from the fact that $(1+\gamma)^u \leq 1 + \gamma u$ and $(1-\gamma)^u \leq 1 - \gamma u$ for any $u \in [0, 1]$ and $\gamma \in [0, 1]$. Note that because the range of rewards is $[0, 1]$, $\lambda_i(x, y) \in [0, 1]$ as well.

Simplifying the equation, we can equivalently rewrite as

$$\gamma \sum_{i=1}^{f(t)} w_{t-1}(i) \left(\lambda_i(x_t, \hat{y}_t) - \lambda_i(x_t, y_t)\right) - \gamma^2 \sum_{i=1}^{f(t)} w_{t-1}(i)\lambda_i(x_t, \hat{y}_t)\lambda_i(x_t, y_t)$$
$$= \gamma \sum_{i=1}^{f(t)} w_{t-1}(i) \left(r_i(x_t, y_t) - r_i(x_t, \hat{y}_t)\right) - \gamma^2 \sum_{i=1}^{f(t)} w_{t-1}(i)\lambda_i(x_t, \hat{y}_t)\lambda_i(x_t, y_t)$$

By definition of our choice of $\hat{y}_t$, the first term is upper bounded by 0, and because all weights and γ values are non-negative, the second term is also upper-bounded by 0, and we conclude that the difference is at most 0, proving the lemma. □

We now proceed to prove Theorem E.5.

Proof of Theorem E.5. We first note that unrolling Lemma E.6 gives a constant upper bound on any W_t:

$$W_t \leq W_{t-1} + \sum_{i=f(t)+1}^{f(t+1)} w_0(i) \leq \sum_{i=1}^{f(t+1)} w_0(i) \leq W.$$

For a particular reward function r_i, the first step where r_i is considered by the algorithm is at $f^{-1}(i) + 1$. We focus on how much additional regret the algorithm can incur after first introducing r_i.

At any timestep $T > f^{-1}(i)$, suppose the algorithm has accumulated $\hat{R}_i(T)$ regret compared to the reward-maximizing labels for r_i since initially being considered by the algorithm at $t = f^{-1}(i) + 1$. Similarly denote $R_i(T)$ as the regret suffered by the demonstrator after timestep $f^{-1}(i)$. By definition of our algorithm, the weight on r_i is equal to

$$w_T(i) = w_0(i)(1+\gamma)^{\hat{R}_i(T)}(1-\gamma)^{R_i(f^{-1}(i)+1:T)} \leq W_T \leq W.$$

Where we get an upper bound of W from Lemma E.6. Solving for $\hat{R}_i(T)$, we conclude that

$$\hat{R}_i(T) \leq \log_2(W/w_0(i))/\log_2(1+\gamma) + R_i(f^{-1}(i)+1:T)\left(\log_2(\frac{1}{1-\gamma})/\log_2(1+\gamma)\right).$$

Noting that for $\gamma \in (0, 3/4]$, we have $\frac{1}{\log_2(1+\gamma)} \leq 1/\gamma$ and $\frac{\log_2(1/(1-\gamma))}{\log_2(1+\gamma)} \leq 1 + 2\gamma$, and thus we conclude that

$$\hat{R}_i(T) \leq \log_2(W/w_0(i))/\gamma + (1+2\gamma)R_i(f^{-1}(i)+1:T).$$

Thus we have bounded the regret after r_i is first considered. We can bound the regret before r_i is first considered as $f^{-1}(i)$ because the rewards lie in $[0, 1]$, and thus conclude a total bound on the regret of

$$M_{i,opt}^T - \hat{M}_{i,alg}^T \leq \log_2(W/w_0(i))/\gamma + (1+2\gamma)R_i(f^{-1}(i)+1:T) + f^{-1}(i).$$

□

We now prove the optimal demonstrator case for binary rewards:

Proof of Theorem E.4. We first note that unrolling Lemma E.6 gives a constant upper bound on any W_t:

$$W_t \leq W_{t-1} + \sum_{i=f(t)+1}^{f(t+1)} w_0(i) \leq \sum_{i=1}^{f(t+1)} w_0(i) \leq W.$$

For a particular reward function r_i, the first step where r_i is considered by the algorithm is at $f^{-1}(i) + 1$. We focus on how much additional regret the algorithm can incur after first introducing r_i.

At any timestep $T > f^{-1}(i)$, suppose there have been M steps where the algorithm outputted a $\hat{y}_t$ with 0 reward on r_i since initially being considered by the algorithm at $t = f^{-1}(i) + 1$. By definition of our algorithm, the weight on r_i is equal to

$$w_T(i) = w_0(i)2^M \leq W_T \leq W.$$

Where we get an upper bound of W from Lemma E.6. Solving for M, we conclude that

$$M \leq \log_2(W/w_0(i)).$$

Thus we have bounded the number of mistakes after r_i is first considered. We can bound the number of mistakes before r_i is first considered as $f^{-1}(i)$, and thus conclude a total bound on the number of mistakes of

$$\sum_{t=1}^{T}(1 - r_i(x_t, \hat{y}_t)) \leq \log_2(W/w_0(i)) + f^{-1}(i).$$

We conclude that this bound holds for all such T, as the right-hand-side has no dependence on T. □