

Language Generation with Infinite Contamination

Anay Mehrotra
Stanford University

ANAYMEHROTRA1@GMAIL.COM

Grigoris Veleghas
Google Research

GVELEGHAS@GOOGLE.COM

Xifan Yu
Yale University

XIFAN.YU@YALE.EDU

Felix Zhou
Yale University

FELIX.ZHOU@YALE.EDU

Editors: Steve Hanneke and Tor Lattimore

Abstract

A recent line of work studies language generation in the limit, a formal model of language learning where an algorithm observes an adversarially generated enumeration of strings from an unknown target language K and must eventually generate new, unseen strings from K . In this model, [Kleinberg and Mullainathan \(2024\)](#) proved that generation is achievable in surprisingly general settings; whenever K belongs to a known countable collection of languages. However, their generator, while quite general, suffers from “mode collapse:” it generates from an ever-smaller subset of the target. To address this, [Kleinberg and Wei \(2025a\)](#) introduced a stronger notion of dense generation, requiring the output to asymptotically cover a positive fraction of the target, and showed it remains achievable for all countable collections.

Both of these works rely on the crucial assumption of *perfect* data: the adversary can neither insert strings from outside the target language (*i.e.*, noise) nor omit strings from it (*i.e.*, omissions). In practice, training data for language models is notoriously noisy, raising the fundamental question:

How much contamination (either omissions or insertions) can language generation tolerate?

Recent works have made partial progress on this question by studying (non-dense) generation with either finite amounts of noise (but no omissions) ([Raman and Raman, 2025](#)) or omissions (but no noise) ([Bai et al., 2026](#)). We characterize the contamination tolerance of both types of generation by proving the following results:

- ▷ **Generation under Contamination:** Language generation in the limit is achievable for all countable collections if and only if the fraction of contaminated examples converges to zero. When this condition fails, we characterize the collections which remain generable.
- ▷ **Dense Generation under Contamination:** Dense generation is achievable for all countable collections if and only if the amount of contamination is finite. For an infinite amount of contamination, we provide several characterizations of when dense generation is possible, showing it is strictly less robust than standard generation.

As a byproduct, we also resolve an open question of ([Raman and Raman, 2025](#)) on generation with membership oracle access under finite contamination.

Keywords: generation, online learning, learning in the limit, robustness

1. Introduction

Large language models (LLMs) have transformed text generation and are already seeing wide-ranging applications from information retrieval to theorem-proving *e.g.*, (Nagda et al., 2025). This progress has sparked a growing body of theoretical work seeking to understand the foundations of language generation (Hanneke et al., 2018; Kalai and Vempala, 2024; Kleinberg and Wei, 2025a; Kalai et al., 2025). We continue this line of inquiry and contribute to incorporating more realistic aspects of language generation in the hope of yielding sharper insights.

The study of language generation in computer science is not new; it predates LLMs and dates back to Turing’s imitation game (Turing, 1950) where he proposed language as a probe of cognition. Most relevant for us is Gold’s model of language identification in the limit, which formulates learning as an online two-player game between an adversary and a learner. In this game, the adversary selects a target language $K = L_z$ from a collection $\mathcal{L} = \{L_1, L_2, \dots\}$, and starts enumerating K one element at a time. After seeing each element, the learner’s goal is to guess the index z of K . The learner is said to succeed if it stabilizes to the right guess z after some finite time (no matter what elements the adversary shows). A line of work, culminating in the works of Angluin (1979, 1980), completely characterized the collections which are identifiable in the limit. This characterization showed that identification is largely intractable; even simple collections like regular languages are not identifiable in the limit.

This raises a puzzle: on the one hand, LLMs acquire non-trivial generation ability from just observing examples; on the other hand, the above model of language learning rules out learnability even for simple collections like regular languages. Kleinberg and Mullainathan (2024) offered an elegant resolution to this puzzle by observing that the requirement of generation is weaker than identification: generation only asks to produce new strings from the language, not to identify the entire language. To formalize this, they introduce a model of language *generation* in the limit:

Informal Definition 1 (Language Generation in the Limit; see Appendix F) *Fix a collection $\mathcal{L} = \{L_1, L_2, \dots\}$. First, the adversary fixes a target language $K \in \mathcal{L}$ and its enumeration.¹ Then, at step $n \geq 1$, it reveals x_n ; given the history $S_n = \{x_1, \dots, x_n\}$, the generator $\mathbb{G}$ outputs a new string $w_n \notin S_n$, which is its guess for an unseen element of K . $\mathbb{G}$ is said to succeed in generating from $\mathcal{L}$ in the limit if for all $K \in \mathcal{L}$ and all enumerations of K , there exists a time n^* such that for every $n \geq n^*$, we have $w_n \in K \setminus S_n$.*

In sharp contrast to identification, Kleinberg and Mullainathan (2024) showed that generation is remarkably feasible: there is a generator that succeeds for *every* countable collection $\mathcal{L}$, even the collection of all Turing-enumerable languages. This surprising positive result sparked a line of work exploring various aspects of language generation, *e.g.*, (Li et al., 2025; Kalavasis et al., 2025; Charikar and Pabbaraju, 2025); see Section 5.

Language Generation with Density. A limitation of the above model is that it does not require meaningful coverage of K . For instance, it accepts a generator that outputs “I generated 1,” “I generated 2,” and so on as a successful generator for English. While this generator indeed demonstrates an interesting ability, it can count, it hardly captures the richness of English. It turns out that Kleinberg and Mullainathan (2024)’s generator exhibits a similar behavior as it operates by generating from progressively “thinner” subsets of K , effectively suffering from mode collapse. To address

1. Formally, an enumeration of K is an infinite sequence $x_1, x_2, \dots$ (possibly with duplicates) such that each $x_i \in K$ and every $x \in K$ appears at some position.

this downside, Kleinberg and Wei (2025a) introduced *language generation with density*, requiring the generator to asymptotically cover a positive “fraction” of K . They proposed several notions of density that, at a high level, are grouped into two categories according to their view of a generator:

- ▷ **Element-based density** focuses on the output sequence generated during the infinite game, without taking into account the “internal” representation of the generator, *e.g.*, the weights of an LLM. This resembles the point of view of a user that is accessing the model through an API; the internal representation of the model might be evolving over time, but the user only sees its sequence of outputs. While we have not defined element-based density formally, we can still gain some intuition: Intuitively, if $K = \mathbb{N}$ and the generator outputs $4\mathbb{N} = \{4, 8, 12, \dots\}$, the element-based density is $1/4$; if K is the even numbers, it becomes $1/2$
- ▷ **Set-based density** measures the density of all elements *producible* by the generator $\mathbb{G}$ at any given time. For this to make sense, $\mathbb{G}$ must be equipped with an option where after each round, we can pause and ask it to generate more and more elements. This is, of course, natural for real-world language models, which once trained are used to generate text as often as necessary. Compared to element-based density, set-based density tries to capture expressivity of one model, rather than of single elements generated by an infinite sequence of models.

Both densities rely on limiting behavior. Element-based density considers an infinite sequence of outputs produced by $\mathbb{G}$. Set-based density considers the limiting behavior of d_t as $t \rightarrow \infty$ where d_t is the density of the generator at the t -th step.² Depending on whether one considers $\liminf$ or $\limsup$ as the notion of a limit, we get four notions of density: lower/upper element-based density and lower/upper set-based density. Since $\liminf d_t \leq \limsup d_t$, lower density is harder to achieve than upper density.

Dense language generation (for any of the aforementioned notions of density) is a substantially stronger goal than generation in the limit because it prevents the generator from retreating to ever-shrinking corners of K . Yet, remarkably, Kleinberg and Wei (2025a) showed strong positive results:

- ▷ **(Element-based Density):** For any countable collection $\mathcal{L}$, there is a generator achieving element-based *lower* density at least $1/8$; and one with element-based *upper* density at least $1/2$.
- ▷ **(Set-based Density):** Set-based lower density is not achievable for all countable collections, but set-based upper density is: there is a generator achieving density 1 for all countable collections.

In this work, we continue this study of dense language generation, bringing it closer to practice.

Language Generation under Contamination. A crucial assumption in Kleinberg and Mullaithan (2024) and most subsequent works is that every element x presented to the generator is valid (*i.e.*, $x \in K$), and, conversely, every (fixed) element $x \in K$ is eventually presented to the generator. This stands in stark contrast to the reality of LLM training. Pre-training data is notoriously noisy and despite extensive resources devoted to data cleaning (Dodge et al., 2021; Raffel et al., 2020) and the emergence of entire industries focused on curating high-quality training data, the web-scale corpora used for pre-training LLMs remain riddled with errors, repetitions, low-quality content, and syntactically invalid code (Lee et al., 2022; Kreutzer et al., 2022). Moreover, while the

2. Set-based density involves two limits: first, a limit over the sequence of elements producible by $\mathbb{G}$ in around t ; which is used to define d_t . Then the limit of d_t as $t \rightarrow \infty$. It turns out that changing the former limit from $\liminf$ to $\limsup$ does not qualitatively change the notion of density; hence, (Kleinberg and Wei, 2025a) fixed the first limit as $\liminf$.

amount of pre-training data keeps increasing, it is unrealistic to hope that it can, eventually, contain every factually correct statement. Yet, modern LLMs consistently generate coherent text despite training on such imperfect data, raising the following question which is the starting point of our work: *Is language generation feasible from training data with large amounts of contamination?*

Two works take an initial step for capturing contamination (Raman and Raman, 2025; Bai et al., 2026). Raman and Raman (2025) allow finitely many invalid examples (but no omissions) and show generation remains possible for all countable collections; however, if the amount of noisy examples is unbounded, then their approach fails. Bai et al. (2026) allow arbitrarily many omissions (without adding any invalid examples) and, again, show generation is possible for all countable collections provided infinitely many valid elements remain. This restriction of no invalid elements, however, is inherent to their approach and their generator fails even with *two* invalid examples (Appendix F.7).³

In practice, both invalid elements and omissions persist simultaneously, and in infinite quantity. Understanding language generation with unbounded quantity of contamination is an important conceptual extension that both brings (Kleinberg and Mullainathan, 2024)’s model closer to reality and motivates new techniques to handle resulting challenges. This raises our main question:

Question. How much *contamination* (both in the form of invalid examples and omissions) can generation and dense generation tolerate while still being possible for all countable collections?

Our results present a complete answer to the above question; we present the formal statements of these results in the next section, but first present a quick summary below.

Summary of Results. We consider different regimes of the amount of contamination in the input stream, and completely characterize when language generation in the limit is achievable under contamination across these regimes. First, we show that language generation shows surprising tolerance: under natural limits on the frequencies of contamination, generation remains possible for all countable collections even with infinite contamination of both types (Theorem 5). In comparison, dense generation is less tolerant: it is possible for all countable collections with finite amounts of noise (and arbitrary omissions) (Theorem 7 and Informal Theorem 8) but becomes impossible for all countable collections with any infinite contamination, no matter how sparse (Table 1). Apart from these main results, we also provide a number of finer-grained characterizations. Our result extends the results of prior works (Raman and Raman, 2025; Bai et al., 2026) in two dimensions: (1) we analyze much more general forms of contamination, and (2) if the amount of tolerance is the same as in prior works, then we can achieve dense generation (for any notion of density) for all countable collections. Interestingly, all our algorithmic results follow from two algorithmic templates (Appendix A), explained in the technical overview (Section 4). We believe these templates can also find applications in other settings of generation in the limit.

Organization. The remainder of the paper is organized as follows. In Section 2 we introduce the formal model of language generation in the limit, define the contamination regimes we study, and review the relevant density notions. In Section 3 we state our main theorems characterizing when (dense and non-dense) generation is achievable under different forms of contamination, and in Section 4 we give a technical overview of the key algorithmic ideas and proof strategies. Finally, Section 5 discusses related work, and Section 6 concludes with discussion and open problems.

3. Both papers study additional variants reviewed in Section 5.

2. Preliminaries

In this section, we introduce the key preliminaries and defer detailed discussion to [Appendix F](#). We use $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ to denote generators. For each n , $S_n = \{x_1, x_2, \dots, x_n\}$ denotes the set of elements revealed by the adversary in the first n iterations and $\mathbb{G}_n(S_n)$ denotes the corresponding output of $\mathbb{G}$. We consider three types of generators:

- ▷ *Element-based* generators, in each step n , output an element $\mathbb{G}_n(S_n) = w_n$,
- ▷ *Set-based* generators, in each step n , output an infinite set $\mathbb{G}_n(S_n) = G_n$.
- ▷ *Index-based*, at each step n , output a language from $\mathcal{L}$, i.e., $\mathbb{G}_n(S_n) = L_{i_n}$.

Index-based generators are a special case of set-based ones and set-based generators can be thought of as an “improper” version of index-based generators; index-based generators were introduced by [\(Kleinberg and Wei, 2025a\)](#) and (improper) set-based generators by [Kalavasis et al. \(2025\)](#).

Definition 1 (Language Generation in the Limit ([Kleinberg and Mullainathan, 2024](#))) Fix collection $\mathcal{L} = \{L_1, L_2, \dots\}$ and a generating algorithm $\mathbb{G} = (\mathbb{G}_n)$. At each step n , let $S_n \subseteq K$ be the set of all strings that $\mathbb{G}$ has seen so far. The generator $\mathbb{G}$ is said to generate from $\mathcal{L}$ in the limit if, for all $K \in \mathcal{L}$ and enumerations of K , there is some $n^* \in \mathbb{N}$ such that for all steps $n \geq n^*$,

1. if $\mathbb{G}$ is element-based, then $\mathbb{G}_n(S_n) \in K \setminus S_n$;
2. if $\mathbb{G}$ is set-based or index-based, then $\mathbb{G}_n(S_n) \subseteq K$.⁴

Next, we define different notions of density from [\(Kleinberg and Wei, 2025a\)](#). For this, we need to fix a canonical enumeration of elements in each language L . We do so by fixing a canonical enumeration of the universe U of all possible strings and letting L^n denote the first n elements in the induced canonical enumeration of L . The two element-based notions of density are as follows:

Definition 2 (Element-based density) Let $W = \{w_1, w_2, \dots\}$ be the sequence of outputs of an element-based generator $\mathbb{G}$ with target K . The element-based lower density and element-based upper density of $\mathbb{G}$ are respectively

$$\mu_{\text{low}}(W; K) := \liminf_n \frac{|W \cap K^n|}{n} \quad \text{and} \quad \mu_{\text{up}}(W; K) := \limsup_n \frac{|W \cap K^n|}{n}.$$

We refer the reader to [Appendix F.1](#) for a refresher on $\liminf$ and $\limsup$, and several examples. Since $\liminf \leq \limsup$ for any sequence, the element-based lower density is harder to satisfy than element-based upper density. Indeed, these two can be very different, e.g., if $W = \cup_{\ell \in \mathbb{N}} [(2\ell)!, (2\ell + 1)!]$, then $\mu_{\text{low}}(W, \mathbb{N}) = 0$ and $\mu_{\text{up}}(W, \mathbb{N}) = 1$. Next we introduce set-based notions of density.

Definition 3 (Set-based density) Let $G = \{G_1, G_2, \dots\}$ be the sequence of sets output by a set-based generator $\mathbb{G}$ with target K . The set-based lower density and set-based upper density of $\mathbb{G}$ are $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(G_n; K)$ and $\limsup_{n \rightarrow \infty} \mu_{\text{low}}(G_n; K)$, respectively.

As before, upper density is a weaker requirement than lower density. To gain some intuition, suppose $G_n = K$ if n is even and, otherwise, $G_n = \{w_i\}$ where w_i is some unseen string in K . Then $\mathbb{G}$ has set-based upper density of 1 and a set-based lower density of 0 (as $\mu_{\text{low}}(\{w_i\}, K) = 0$).

The final ingredient we need to state our results is the levels of contamination considered.

Definition 4 (Contamination Regimes) Fix a language K and an enumeration $E = (x_1, x_2, \dots)$ not necessarily of K . We focus on the following four cases:

4. Note that if a set-based generator satisfies $\mathbb{G}_n(S_n) \subseteq K$, then we can modify it to satisfy the stronger requirement $\mathbb{G}_n(S_n) \subseteq K \setminus S_n$. We use the weaker requirement to simplify presentation.

1. E is an enumeration of K with **finite contamination** if $|E \triangle K| < \infty$.⁵
2. E is an enumeration of K with **$o(1)$ -Noise** if $|\{x_1, \dots, x_n\} \setminus K|/n = o(1)$.
3. E is an enumeration of K with **c -noise** if $|\{x_1, \dots, x_n\} \setminus K|/n \leq c$, for all $n \geq n^*$ for some finite n^* .
4. E is an enumeration of K with **arbitrary omissions** if $|K \setminus E| = \infty$ (and $|E \cap K| = \infty$).

Here, finite contamination allows both invalid examples and omissions, and already generalize the case of finite noise from [Raman and Raman \(2025\)](#). The next two regimes quantify the fraction of noisy examples among the first n : in $o(1)$ -noise this fraction tends to 0, and in c -noise, it becomes $\leq c$ for large enough n . With omissions, quantifying “the fraction of omissions” is more nuanced since missing elements could always appear later. Hence, we primarily focus on two regimes: finite omissions (a special case of finite contamination) and arbitrary omission (studied by [Bai et al. \(2026\)](#) without noise). Later, we will consider a notion of **c -omissions**, requiring $\mu_{\text{low}}(K, E) \geq 1 - c$.

3. Our Results

Our first result studies generation under contamination.

Theorem 5 (Generation with Contamination) *There is a generator that, for any countable language collection $\mathcal{L}$, generates in the limit from $\mathcal{L}$ under $o(1)$ -noise and arbitrary omissions.*

Thus, generation is quite tolerant to contamination: even with infinite contamination, it remains possible for all countable collections when the noise-fraction tends to 0. This significantly generalizes both [Raman and Raman \(2025\)](#); [Bai et al. \(2026\)](#). It might seem surprising that we impose no constraint on omissions; this stems from generation’s asymmetric objective: it requires generating new unseen elements, thus penalizing false positives more than false negatives. Our generator utilizes a novel idea: it carefully re-orders languages based on the adversary’s actions, contrasting with existing generators that use static orderings ([Section 4](#)). Further, it outputs (elements from) intersections of certain languages and, hence, is not an index-based generator; we show that this is necessary ([Remark 10](#) and [Example 8](#)). The formal proof is deferred to [Appendix B.1](#). Next, we show the requirement of $o(1)$ -noise is necessary.

Informal Theorem 6 (Generation with c -contamination; see [Theorem 16](#)) *Fix any $c \in (0, 1)$. Under c -noise and arbitrary omissions, a countable language collection $\mathcal{L}$ is generable in the limit if and only if $\mathcal{L}$ satisfies the condition in [Theorem 16](#) with parameter c .*

Moreover, for each $c \in (0, 1)$, there is a finite collection $\mathcal{L}$ that violates this condition.

This result reveals the limits of generation’s contamination tolerance. While we omit the technical details of the condition, we note that all finite collections of size at most $1/c$ satisfy the condition with parameter c . This generator also uses the idea of carefully reordering languages. The proof of this result can be found in [Appendix B.2](#).

Dense Generation with contamination. Next, we turn to dense generation.

Theorem 7 (Dense Generation with Contamination I) *Under finite contamination and for any countable collection $\mathcal{L}$: (i) there is a generator achieving set-based upper density 1; (ii) there is a generator achieving element-based upper density $\rho = 1/2$; and (iii) there is a generator achieving element-based lower density $\rho = 1/8$.*

5. If we want to distinguish between the two types of contaminations, we say, **finite noise** (denoting $|E \setminus K| < \infty$) and **finite omissions** (denoting $|K \setminus E| < \infty$).

Informal Theorem 8 (Dense Generation with Contamination II; see Theorem 23) *Under finite contamination, a countable collection $\mathcal{L}$ is generable in the limit with ρ set-based lower density if and only if $\mathcal{L}$ satisfies the condition in Theorem 23 with parameter ρ .*

Thus, all notions of dense generation except set-based lower density are tolerant to finite amounts of contamination. Moreover, all of these results are tight: dense generation (for any of the aforementioned notions) cannot be achieved for all countable collections if the amount of contamination is infinite, no matter how sparse (Table 1). The proofs of these results appear in Appendices C.1 and C.2, respectively. Some remarks are in order:

- ▷ Set-based lower density is particularly challenging: even under finite contamination, it is unachievable whenever $\mathcal{L}$ has two languages P, L with $P \subseteq L$ and $\mu_{\text{low}}(P, L) = 0$ (e.g., $L = \mathbb{N}$ and P is primes) (Informal Theorem 8). This is far more stringent than the contamination-free case, where (Kleinberg and Wei, 2025a) showed impossibility requires a pathological sub-collection of infinite size.
- ▷ For parts (ii) and (iii) in Theorem 7, we actually prove something stronger: any algorithm achieving element-based density ρ (for all countable collections) without contamination yields one achieving the same ρ under finite contamination (Theorem 21). Hence, remarkably, element-based density with finite contamination is exactly as hard as with no contamination, contrasting with set-based lower density which became significantly harder. (Indeed, the bounds $1/2$ and $1/8$ match (Kleinberg and Wei, 2025a)’s contamination-free results.)

Remark 8 *The set-based upper density result is stronger: it tolerates finite noise with infinite omissions, provided omissions are not too dense (see c -omissions in Section 2 and Theorems 17 and 18).*

As we mentioned before, all our algorithms above, broadly speaking, rely on our two algorithmic ideas: all generators for finite contamination use a finite expansion subroutine and all other algorithms use the idea of carefully re-ordering languages based on the adversary’s actions. We believe these ideas can be of independent interest in the study of language generation. As an illustration, we use the finite expansion subroutine to answer an open question from (Raman and Raman, 2025):

Corollary 9 (Membership-Query–Based Algorithm with Finite Contamination) *There is a computable $\mathbb{G}$ that, for any countable collection $\mathcal{L}$, given access to an oracle which given w and i answers “Is $w \in L_i$?”, generates $\mathcal{L}$ in the limit under finite contamination.*

In contrast, Raman and Raman (2025) require more oracles to handle finite noise (and 0 omission). The description of the algorithmic template and the proof this result can be found in Appendix A.2.

Further Results for Dense Generation. Beyond the results above, we provide fine-grained characterizations of dense generation under $o(1)$ -noise and c -noise regimes. These results, summarized in Table 1, characterize which countable collections admit dense generation under various contamination models. Notably, two characterizations remain open: while we establish impossibility for arbitrary countable collections (Theorem 32), complete characterizations for these cases are interesting directions. We also observe several qualitative phenomena. First, while set-based lower and upper densities have distinct characterizations without contamination (Kleinberg and Wei, 2025a), they collapse to a single characterization under contamination. Second, as noted before, element-based density is notably more contamination-tolerant than set-based density (Informal Theorem 8).

Objective	Contamination Regime		
	Finite Contamination	$o(1)$ -Noise and Arbitrary Omissions	c -Noise and Arbitrary Omissions
Generation	Theorem 5 or 7 (FE)	Theorem 5 (PI)	Theorem 16 (PI) (C)
Set-Based Upper	Theorem 7 (FE) ($\star$)	Theorem 27 (PI) (C)	Theorem 28 (PI) (C)
Set-Based Lower	Theorem 7 (FE) (C)	Theorem 27 (PI) (C)	Theorem 28 (PI) (C)
Element-Based	Theorem 7 (FE) ($\star$)	Open (Theorem 32)	Open (Theorem 32)

Table 1: *Summary of results across contamination regimes.* All results use two broad algorithmic ideas or templates: a priority-based intersection meta-algorithm (PI; [Appendix A.1](#)) and a finite expansion subroutine (FE; [Appendix A.2](#)). **Legend:** (C) = characterization of which collections achieve the guarantee (otherwise achievable for all countable collections); (R) = reduction from set-based to element-based density ([Theorem 23](#)); Open = impossibility shown for all countable collections ([Theorem 32](#)) but full characterization remains open.

Note ($\star$) This result is stronger: it tolerates finite noise with *infinite* omissions, provided omissions are not too dense (c -omissions; [Appendix F](#)). See [Theorems 17](#) and [18](#).

A general takeaway is that dense generation is much less contamination-tolerant than non-dense generation, contrasting ([Kleinberg and Wei, 2025a](#))’s positive results, which suggested density was achievable whenever generation was.

Beyond Worst-Case Adversaries ([Appendix D](#)). We also study a restricted adversarial model inspired by curriculum learning ([Bengio et al., 2009](#)): An adversary is M -bounded if, for sufficiently large n , any $x_n \in K$ it enumerates satisfies $x_n = \kappa_i$ for some $i \leq Mn$, where $\kappa_1, \kappa_2, \dots$ is the canonical enumeration of K . Intuitively, if we order the elements of the language starting from easier ones to harder ones, then such adversaries cannot present “hard” elements far before “easier” ones. Under this model, dense generation becomes significantly more tractable: all countable collections admit set-based and element-based lower density $\Omega(1/M)$ under $o(1)$ -noise and arbitrary omissions, and these bounds are tight up to a factor of 2 ([Theorems 39](#) and [40](#)). This contrasts sharply with arbitrary adversaries, where no density notion is universally achievable even under arbitrarily sparse infinite contamination.

Remark 10 (Proper vs. Improper Learning in the Worst-Case) *Our set-based algorithms are improper in the sense they output sets which are not part of the original collection $\mathcal{L}$: in particular, they output intersections of finitely many languages from $\mathcal{L}$, rather than a single $L \in \mathcal{L}$. This is necessary: [Example 8](#) shows proper generation under contamination is far more restrictive. This contrasts with the uncontaminated setting, where proper and improper generators are equally powerful ([Kleinberg and Mullainathan, 2024](#); [Kleinberg and Wei, 2025a](#)).*

4. Technical Overview

In this section, we give an overview of our techniques and how they relate to prior works. We divide this into two parts: upper bounds (or algorithms) and lower bounds.

4.1. Upper Bounds for Dense and Non-Dense Generation

All of our algorithms across generation with and without density, rely on two algorithmic ideas or *templates*. Here, we overview these templates. They are discussed in more detail in [Appendix A](#).

Finite-Expansion Sub-Routine (Appendix A.2). Our first algorithmic template is quite simple and enables us to achieve dense and non-dense generation with finite contamination. The algorithm relies on the following elementary observation: If the enumeration E generated by the adversary has finite contamination, then it must be a 0-contamination enumeration of a language K' satisfying with $|K' \triangle K| < \infty$. Now, of course, K' might not be a language in our collection $\mathcal{L}$, but the natural thing to do is to add K' to $\mathcal{L}$. We will need to add all possible choices since we do not know K' in advance. Then, we end up with the following collection

$$\widetilde{\mathcal{L}} := \{L_{A,B} := L \cup A \setminus B : L \in \mathcal{L}, A \subseteq \Sigma^* \setminus L, B \subseteq L, |A| < \infty, |B| < \infty\},$$

where Σ^* is the universe of strings. Since the number of finite subsets of a countable collection are countable, $\widetilde{\mathcal{L}}$ is a countable collection. This immediately gives us an algorithm for generation with finite amounts of contamination since we can use (Kleinberg and Mullainathan, 2024)’s algorithm that works for all countable collections with $\widetilde{\mathcal{L}}$ instead of $\mathcal{L}$. This simple idea yields clean analyses while offering many useful properties that make it widely applicable; including but not limited to:

- ▷ As we have seen, it preserves countability (if $\mathcal{L}$ is countable, so is $\widetilde{\mathcal{L}}$).
- ▷ Further, membership access to $\mathcal{L}$ is sufficient to get membership⁶ access to $\widetilde{\mathcal{L}}$.
- ▷ In several contamination regimes and density measures, if G achieves density with respect to $\mathcal{L}$ it also achieves density with respect to $\widetilde{\mathcal{L}}$.

This transformation also preserves generation with approximate breadth, which is a stronger notion than density introduced in Kalavasis et al. (2025). We leverage such properties in our results, and we believe there may be additional benefits. For instance, points 1 and 2 above together allow us to get a membership oracle-based algorithm for generation with finite contamination (Corollary 9) that resolves an open question from Raman and Raman (2025); it is unclear how to resolve it using using prior techniques.

Priority-Based Intersection Algorithm (Appendix A.1). Next, we discuss our algorithms for the much more involved case where the amount of contamination is infinite. We begin by explaining why earlier approaches fail:

- ▷ The previous simple approach that constructs the “expanded” collection $\widetilde{\mathcal{L}}$ fails because the collection $\mathcal{L}$ becomes uncountable (if the amount of noise is not finite) and not all uncountable collections are generable in the limit, even without the requirement of density. The situation with density is even more complicated as the transformation is no longer density preserving.
- ▷ Another idea is to use approaches from Raman and Raman (2025); Bai et al. (2026). Raman and Raman (2025) use a nice observation for generating in the limit: if the amount of noise is finite then, for sufficiently large n , the second half of the training examples (namely, $\{x_{n/2}, x_{n/2+1}, \dots, x_n\}$) eventually contains no noisy examples. Hence, roughly speaking, feeding the second half of the examples to an appropriate generation algorithm suffices. This, of course, fails when there is an infinite amount of noisy examples in the stream. Bai et al. (2026) make another nice observation: if G has the property that it generates from $\mathcal{L}$ after $n^* = n^*(K)$ iterations, where n^* depends on the target K , but not on its enumeration, then G generates from $\mathcal{L}$ in the limit under arbitrary omissions (provided there is no noise). This suffices as Charikar and Pabbaraju (2025) constructed a generator with this property for all

6. A membership oracle to $\mathcal{L} = \{L_1, L_2, \dots\}$ is a primitive that, given $w \in \Sigma^*$ and i , answers “Is $w \in L_i$?”

countable collections. However, this approach fails even when there are two noisy examples in the enumeration without any omission (see [Appendix F.7](#)).

To understand our (meta) algorithm, it is instructive to first understand the algorithm of [Charikar and Pabbaraju \(2025\)](#). Roughly speaking, in the n -th iteration, their algorithm considers the n languages $\{L_1, L_2, \dots, L_n\}$ and, from these, it removes any language inconsistent with the training data seen so far S_n , i.e., any $L \not\supseteq S_n$. Let the resulting languages be $L_{\sigma(1)}, L_{\sigma(2)}, \dots, L_{\sigma(m)}$ (for $m \leq n$). Then, it initializes $O_t \leftarrow \Sigma^*$ and traverses the list $L_{\sigma(1)}, \dots, L_{\sigma(\ell)}$. If $|O_t \cap L_{\sigma(j)}| = \infty$ it updates $O_t \leftarrow O_t \cap L_{\sigma(j)}$, otherwise it proceeds to the next element without changing O_t .

Of course, this algorithm does not directly work when there is contamination in the sample stream because, for instance, the consistency check of $L \not\supseteq S_n$ is no longer meaningful. Consider the case of c -noise; let the enumeration be $E = \{x_1, x_2, \dots\}$. The natural counter part is to check whether E is an c -noisy enumeration of L . This, however, requires having access to the entire enumeration E , which we have never seen at any iteration $n < \infty$. One could instead check if the fraction of elements from $\{x_1, x_2, \dots, x_n\}$ not in L , which we term the *empirical noise rate*, is at most $c + \varepsilon$ for small fixed $\varepsilon > 0$. There are examples where this fails because this approach can include “bad” languages L that do not meet the actual requirement that E is a c -noisy enumeration for L . One could tighten this check by setting $\varepsilon = 0$, but then we run into the issue that the empirical noise rate for certain bad languages can fluctuate above and below c infinitely often. To gain some intuition suppose $K = \{2, 4, 6, \dots\}$ and $L = \{n : n \in \mathbb{N}, n \bmod 4 \neq 0\} \cap A$ where $A = \bigcup_{\ell \in \mathbb{N}} [(2\ell)!, (2\ell + 1)!]$. Then, the enumeration $E = \{1, 2, 3, \dots\}$, is a $1/2$ -noisy enumeration for K and $3/4$ -noisy enumeration of L (so L should be excluded in our check). However, the empirical noise rate of L fluctuates between 0 and $3/4$ infinitely often (and, hence, L would end up being included in our check infinitely often).

This is not merely a superficial problem in the above approaches, but rather an inherent problem in verifying c -noise or $o(1)$ -noise at finite times — which is impossible as they are necessarily limiting phenomena. Hence, to overcome this we need a new approach that is able to (i) remove “bad” languages from our “active set” of languages, and (ii) ensure that the target language K always remains in this set after some finite time. To design our approach, we take inspiration from the failure of the above approaches, where certain “bad” languages fluctuate between passing our check and failing our check infinitely often. Our algorithm assigns each language a *priority*. Initially, the priority of language L_i is simply $-i$ (so the languages in order of priority are $L_1, L_2, \dots$), and each time L_i fails our check we penalize the language by decreasing its priority. The key observation is that even though K might be getting penalized infinitely often, it will, eventually, be part of a *stable prefix*. Then, taking intersections of prefixes, as long as these remain infinite ensures that in the limit, we output subsets of K . Similar properties of the orderings induced by priorities enable us to get appropriate notions of density.

Like our previous algorithmic template, this also has quite general applicability: to all notions of density and also for generation with bounded adversaries. Indeed, this core algorithmic template of assigning priorities to languages shows up in all of our algorithms beyond the finite contamination regime, each time, with a slightly different notion of priority. For instance, with M -bounded adversaries, we need to check whether the provided enumeration E is indeed M -bounded for a language L ; this is again a limiting phenomenon which cannot be checked at any finite time. The situation for dense generation is even more complicated. Indeed, [Kleinberg and Wei \(2025a\)](#) already demonstrated that achieving lower density (both element-based set-based) for all countable collections is very difficult and their algorithms carefully need to balance the trade-off between generating only

valid elements and covering enough fraction of K . Now, with infinite amount of contamination this becomes even harder due to the fluctuating empirical noise rates we discussed above.

Finally, we provide another view that connects our priority-based intersection algorithm with ERM (empirical risk minimization). For every language L_i , consider a natural “loss” $\ell_t(i)$ at time t that measures the fraction of the elements of the stream up to time t that are not contained in L_i . In all of our applications of the priority-based intersection algorithm, the priority score we assign to each language is a function of both the “loss” term and a “regularization” term, similar to the *regularized* ERM. However, the priority-based intersection algorithm diverges from regularized ERM in the output rule. Instead of outputting a single language with the minimum “risk”, the priority-based intersection algorithm outputs from the intersection of a carefully chosen collection of languages with the lowest risks.

4.2. Lower Bounds for Dense and Non-Dense Generation

A common theme in learning theory is that, once we have the “right” algorithm, we can obtain tight lower bounds by understanding when the algorithm fails. This is also the case with our lower bounds, which are derived by first understanding the obstructions to generability (with different notions of density and contamination) and then formalizing them using diagonalization arguments. An interesting property of our lower bounds is that many of them have a finite witness. For instance, consider dense generation with finite contamination. All our characterizations here have the following form: A countable collection $\mathcal{L}$ violates the condition if there are two pathological languages $L_1, L_2 \in \mathcal{L}$ with a certain property. (For example, that $L_1 \subseteq L_2$ and $\mu_{\text{low}}(L_1, L_2) = 0$.) This contrasts existing lower bounds in generation and identification which required infinite witnesses. For instance, [Kleinberg and Wei \(2025a\)](#) showed that the existence of an infinite perfect tower characterizes lower set-based density. However, as its name suggests, an infinite perfect tower is necessarily witnessed by an infinite sub-collection of languages. Similarly, the characterizations of identification in the limit ([Angluin, 1980](#)) and language generation with breath ([Kalavasis et al., 2025](#); [Charikar and Pabbaraju, 2025](#); [Kalavasis et al., 2026](#)) also require infinite witnesses.

5. Further Related Work

5.1. Related Work on Language Identification in the Limit

Starting from [Gold \(1967\)](#) there has been a rich line of work including both linguistics and computer science on the model of language generation in the limit. Of particular relevance to our work are the line of works on language identification in the limit in settings where the adversary can include invalid examples in the enumeration (*i.e.*, noise) or omit elements ([Mukouchi and Sato, 2003b](#); [Case et al., 1997](#); [Stephan, 1997](#); [Baliga et al., 1992](#); [Fulk and Jain, 1989](#); [Schäfer, 1985](#)). In particular, [Jain \(1994\)](#) extends Gold’s model to streams with infinitely many inaccuracies, controlling corruption via density (covering finite, vanishing, and constant-rate noise). [Mukouchi and Sato \(2003a\)](#) use a neighbor-system (metric) model that permits insertions and deletions provided each corrupted item lies within a fixed distance of some true string. [Tantini et al. \(2006\)](#) analyze the same metric-based setting and prove identification results under such systematic noise.

5.2. Related Work on Language Generation in the Limit

Next, we discuss works on language generation in the limit. We have already discussed some of these works in the introduction. We discuss the key related works here, and then defer a detailed and thorough discussion of other works to [Appendix G](#).

Language Generation in a Statistical Model. Apart from noise, Kleinberg and Mullainathan (2024)’s model has been extended along another axis: the enumeration of examples. While Kleinberg and Mullainathan (2024) study generation under worst-case adversarial enumeration, motivated by Bousquet et al. (2021), Kalavasis et al. (2025) introduce a statistical model where samples are drawn i.i.d. from a fixed distribution $\mathcal{P}$ supported on K . In this model, they obtain the first and near-tight sample complexities of identification in the limit, generation in the limit, and related tasks. Apart from (Bousquet et al., 2021), (Kalavasis et al., 2025)’s model was also described by Angluin (1988) in the context of language identification. This stochastic model is incomparable to the worst-case to the bounded displacement model we study here. It is an interesting question to understand if the stronger results we obtain for dense generation in the bounded-adversary model can also be obtained in the statistical model.

Noise and Omissions. As already mentioned in Section 1, the two most relevant works are Raman and Raman (2025) and Bai et al. (2026). Raman and Raman (2025) initiated the study of language generation with a *finite* amount of additive noise n^* (unknown to the generator), characterizing the classes that are uniformly generable in this setting. In particular, their results imply that all countable collections remain generable with finite additive noise. Bai et al. (2026) show that infinite omissions do not change which collections are uniformly or non-uniformly generable; combined with Charikar and Pabbaraju (2025), this implies that all countable collections remain non-uniformly generable with infinite omissions. They also study the finite-additive-noise setting of Raman and Raman (2025), providing a characterization of non-uniform generation in that regime. The main additional challenges in our work are: (i) handling an *infinite* amount of additive noise, where the algorithms from these prior works fail, and (ii) simultaneously handling both additive noise and omissions.

Breadth and Density. Prior to the density-based definitions of “breadth” proposed by Kleinberg and Wei (2025a), the parallel works of Kalavasis et al. (2025); Charikar and Pabbaraju (2025); Kalavasis et al. (2026) introduced more stringent notions, requiring (roughly) that the set of elements the generator can produce at each timestep eventually becomes a subset of K missing only *finitely* many elements. They characterized when such breadth is achievable. Moreover, Kalavasis et al. (2025) introduced and studied a statistical model and Charikar and Pabbaraju (2025) explored role of “feedback” and non-uniform generation, with different types of oracles. (Non-uniform generation studies how quickly the learner can start outputting valid, unseen elements of K .) Separately, Peale et al. (2025) introduced “representative generation,” where strings are partitioned into groups and the generator must eventually represent all groups; a relaxation of the breadth notions above.

Concurrent and Independent Work. Independent and concurrent work by Kleinberg and Wei (2025b) also studies dense language generation with omissions but without noise. Their main result is a generator with element-based lower density $(1-c)/2$ under c -omissions (Appendix F), which is tight. Our works share some qualitative ingredients: both use a generator that infinitely often outputs a set containing all observed examples and contained in K , and both observe that proper (index-based) learning fails under omissions. Beyond this, the works diverge: Kleinberg and Wei (2025b) optimize density under omissions, while we handle both noise and omissions across various regimes (Appendix A.1) and study a beyond-worst-case model. Their tight density bounds also have implications for our results, improving Theorem 7 from $1/8$ to $1/2$. See Appendix F.8 for details.

5.3. Other Theoretical Efforts at Understanding Language Generation

Beyond the works discussed above, we briefly highlight a few other theoretical directions on understanding LLMs. A long line of work studies the representational power of transformer archi-

tructures; see [Sanford et al. \(2023\)](#); [Peng et al. \(2024\)](#); [Chen et al. \(2024\)](#) and references therein. Recent efforts have also provided theoretical evidence for the effectiveness of chain-of-thought prompting ([Malach, 2023](#); [Altabaa et al., 2025](#); [Joshi et al., 2025](#); [Huang et al., 2025](#)). On the topic of hallucinations, several works have shown that LLMs providing non-trivial utility *must* hallucinate ([Kalai and Vempala, 2024](#); [Kalai et al., 2025](#); [Xu et al., 2024](#)). Finally, [Liu and Moitra \(2025\)](#), building on [Mahajan et al. \(2023\)](#), showed that *model stealing*, inferring the parameters of a proprietary LLM through API calls, is possible for any low-rank LLM under a theoretical model.

6. Discussion and Open Problems

In this work, we characterized the contamination tolerance of language generation in the limit ([Theorems 5 and 7](#) and [Informal Theorem 8](#)). We showed that standard generation can tolerate contamination in infinite quantity, only requiring that the fraction of contaminated examples converges to zero. Dense generation, in contrast, can only tolerate finite amounts of contamination. Thus, the additional coverage guarantees of dense generation come at a cost in robustness.

There are several interesting future directions related to noisy generation, and more broadly, the line of work on generation in the limit. An interesting direction is to fully characterize element-based generation for all types of contamination model we study. Further, our finite-contamination algorithm uses only membership oracle access to $\mathcal{L}$, but several of our other algorithms require more powerful oracles; understanding what can be achieved with simpler oracles remains open. Finally, beyond restricting the adversary, another way to circumvent our lower bounds may be to relax the learner’s requirements: how does the landscape of (noisy) generation change if the generator may output a vanishing fraction of hallucinations? We note that a similar question was posed by [Kleinberg and Wei \(2025a\)](#) in the context of improving density guarantees.

Acknowledgments

Felix Zhou acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). Xifan Yu is supported in part by ONR Award N00014-24-1-2611.

References

- Awni Altabaa, Omar Montasser, and John Lafferty. Cot information: Improved sample complexity under chain-of-thought supervision. *arXiv preprint arXiv:2505.15927*, 2025.
- Dana Angluin. Finding patterns common to a set of strings (extended abstract). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC ’79, page 130–141, New York, NY, USA, 1979. Association for Computing Machinery. ISBN 9781450374385. doi: 10.1145/800135.804406. URL <https://doi.org/10.1145/800135.804406>.
- Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980. ISSN 0019-9958. doi: [https://doi.org/10.1016/S0019-9958\(80\)90285-5](https://doi.org/10.1016/S0019-9958(80)90285-5). URL <https://www.sciencedirect.com/science/article/pii/S0019995880902855>.
- Dana Angluin. *Identifying Languages From Stochastic Examples*. Yale University. Department of Computer Science, 1988. URL <http://www.cs.yale.edu/publications/techreports/tr614.pdf>.

- Yannan Bai, Debmalya Panigrahi, and Ian Zhang. Language generation in the limit: Noise, loss, and feedback. In *Proceedings of the 2026 ACM–SIAM Symposium on Discrete Algorithms (SODA’26)*. SIAM, 2026. URL <https://arxiv.org/abs/2507.15319>. To appear.
- Ganesh Baliga, Sanjay Jain, and Arun Sharma. Learning from multiple sources of inaccurate data. In *International Workshop on Analogical and Inductive Inference*, pages 108–128. Springer, 1992.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- Olivier Bousquet, Steve Hanneke, Shay Moran, Ramon van Handel, and Amir Yehudayoff. A theory of universal learning. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 532–541, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380539. doi: 10.1145/3406325.3451087. URL <https://doi.org/10.1145/3406325.3451087>.
- John Case, Sanjay Jain, and Arun Sharma. Synthesizing noise-tolerant language learners. In *International Workshop on Algorithmic Learning Theory*, pages 228–243. Springer, 1997.
- Moses Charikar and Chirag Pabbaraju. Exploring facets of language generation in the limit. In Nika Haghtalab and Ankur Moitra, editors, *Thirty-eighth Conference on Learning Theory (COLT 2025)*, Proceedings of Machine Learning Research. PMLR, 2025. URL <https://arxiv.org/abs/2411.09642>.
- Moses Charikar and Chirag Pabbaraju. Pareto-optimal non-uniform language generation. In Matus Telgarsky and Jonathan Ullman, editors, *Proceedings of the 37th International Conference on Algorithmic Learning Theory (ALT 2026)*, Proceedings of Machine Learning Research, Toronto, Canada, February 2026. doi: 10.48550/arXiv.2510.02795. URL <https://arxiv.org/abs/2510.02795>. Accepted to ALT 2026.
- Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer transformer, 2024. URL <https://arxiv.org/abs/2412.02975>.
- Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.98. URL <https://aclanthology.org/2021.emnlp-main.98/>.
- Mark Fulk and Sanjay Jain. Learning in the presence of inaccurate information. 1989.
- E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967. ISSN 0019-9958. doi: [https://doi.org/10.1016/S0019-9958\(67\)91165-5](https://doi.org/10.1016/S0019-9958(67)91165-5). URL <https://www.sciencedirect.com/science/article/pii/S0019995867911655>.
- Guy Hacothen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *International conference on machine learning*, pages 2535–2544. PMLR, 2019.

- Steve Hanneke, Adam Tauman Kalai, Gautam Kamath, and Christos Tzamos. Actively avoiding nonsense in generative models. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 209–227. PMLR, 07 2018. URL <https://proceedings.mlr.press/v75/hanneke18a.html>.
- Steve Hanneke, Amin Karbasi, Anay Mehrotra, and Grigoris Velegkas. On union-closedness of language generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=6h7HLx1kbH>.
- Jianhao Huang, Zixuan Wang, and Jason D Lee. Transformers learn to implement multi-step gradient descent with chain of thought. *arXiv preprint arXiv:2502.21212*, 2025.
- Sanjay Jain. Program synthesis in the presence of infinite number of inaccuracies. In Setsuo Arikawa and Klaus P. Jantke, editors, *Algorithmic Learning Theory*, pages 333–348, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. ISBN 978-3-540-49030-2.
- Nirmit Joshi, Gal Vardi, Adam Block, Surbhi Goel, Zhiyuan Li, Theodor Misiakiewicz, and Nathan Srebro. A theory of learning with autoregressive chain of thought. *arXiv preprint arXiv:2503.07932*, 2025.
- Adam Tauman Kalai and Santosh S. Vempala. Calibrated language models must hallucinate. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 160–171, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703836. doi: 10.1145/3618260.3649777. URL <https://doi.org/10.1145/3618260.3649777>.
- Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate, 2025. URL <https://arxiv.org/abs/2509.04664>.
- 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 (STOC’25)*, New York, NY, USA, 2025. Association for Computing Machinery. URL <https://arxiv.org/abs/2411.09642>.
- Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. On characterizations for language generation: Interplay of hallucinations, breadth, and stability. In Matus Telgarsky and Jonathan Ullman, editors, *Proceedings of the 37th International Conference on Algorithmic Learning Theory (ALT 2026)*, *Proceedings of Machine Learning Research*, Toronto, Canada, February 2026. doi: 10.48550/arXiv.2412.18530. URL <https://arxiv.org/abs/2412.18530>. Accepted to ALT 2026.
- Amin Karbasi, Omar Montasser, John Sous, and Grigoris Velegkas. (im)possibility of automated hallucination detection in large language models. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=e5jWdZIX0Q>.
- 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. In *Proceedings of the 66th IEEE Symposium on Foundations of Computer Science (FOCS 2025)*. IEEE, 2025a. URL <https://arxiv.org/abs/2504.14370>. To appear.
- Jon Kleinberg and Fan Wei. Language generation and identification from partial enumeration: Tight density bounds and topological characterizations, 2025b. URL <https://arxiv.org/abs/2511.05295>.
- Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmongkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroko Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhlov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Balı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72, 2022. doi: 10.1162/tacl_a_00447. URL <https://aclanthology.org/2022.tacl-1.4/>.
- Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.577. URL <https://aclanthology.org/2022.acl-long.577/>.
- Jiaxun Li, Vinod Raman, and Ambuj Tewari. Generation through the lens of learning theory. In Nika Haghtalab and Ankur Moitra, editors, *The Thirty Eighth Annual Conference on Learning Theory, 30-4 July 2025, Lyon, France*, volume 291 of *Proceedings of Machine Learning Research*, pages 4740–4776. PMLR, 2025. URL <https://proceedings.mlr.press/v291/raman25a.html>.
- Allen Liu and Ankur Moitra. Model stealing for any low-rank language model. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 1755–1761, 2025.
- Gaurav Mahajan, Sham Kakade, Akshay Krishnamurthy, and Cyril Zhang. Learning hidden markov models using conditional samples. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 2014–2066. PMLR, 2023.
- Eran Malach. Auto-regressive next-token predictors are universal learners. *arXiv preprint arXiv:2309.06979*, 2023.
- Yasuhito Mukouchi and Masako Sato. Refutable language learning with a neighbor system. *Theoretical Computer Science*, 298(1):89–110, 2003a. ISSN 0304-3975. doi: [https://doi.org/10.1016/S0304-3975\(02\)00420-6](https://doi.org/10.1016/S0304-3975(02)00420-6). URL <https://www.sciencedirect.com/science/article/pii/S0304397502004206>. Selected Papers in honour of Setsuo Arikawa.

- Yasuhito Mukouchi and Masako Sato. Refutable language learning with a neighbor system. *Theoretical computer science*, 298(1):89–110, 2003b.
- Ansh Nagda, Prabhakar Raghavan, and Abhradeep Thakurta. Reinforced generation of combinatorial structures: Applications to complexity theory, 2025. URL <https://arxiv.org/abs/2509.18057>.
- Charlotte Peale, Vinod Raman, and Omer Reingold. Representative language generation. In *Forty-second International Conference on Machine Learning*, 2025.
- Binghui Peng, Srini Narayanan, and Christos Papadimitriou. On limitations of the transformer architecture. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=KidynPuLNW>.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified Text-to-Text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <https://jmlr.org/papers/v21/20-074.html>.
- Ananth Raman and Vinod Raman. Generation from noisy examples. In *Forty-second International Conference on Machine Learning*, 2025.
- Walter Rudin. *Principles of Mathematical Analysis*. McGraw–Hill, 3 edition, 1976.
- Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 36677–36707. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/73bf692447f174984f30499ec9b20e04-Paper-Conference.pdf.
- Gisela Schäfer. Some results in the theory of effective program synthesis: Learning by defective information. In *Int. Spring School on Mathematical Methods of Specification and Synthesis of Software Systems*, pages 219–225. Springer, 1985.
- Frank Stephan. Noisy inference and oracles. *Theoretical Computer Science*, 185(1):129–157, 1997.
- Frédéric Tantini, Colin de la Higuera, and Jean-Christophe Janodet. Identification in the limit of systematic-noisy languages. In Yasubumi Sakakibara, Satoshi Kobayashi, Kengo Sato, Tetsuro Nishino, and Etsuji Tomita, editors, *Grammatical Inference: Algorithms and Applications*, pages 19–31, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-45265-2.
- Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950. doi: 10.1093/MIND/LIX.236.433. URL <https://doi.org/10.1093/mind/LIX.236.433>.
- Keyon Vafa, Justin Y. Chen, Ashesh Rambachan, Jon M. Kleinberg, and Sendhil Mullainathan. Evaluating the world model implicit in a generative model. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural*

Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/2f6a6317bada76b26a4f61bb70a7db59-Abstract-Conference.html.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of large language models, 2024. URL <https://arxiv.org/abs/2401.11817>.

Organization of the Appendix

We briefly summarize the contents of each appendix.

- ▷ [Appendix A](#) presents the algorithmic templates used throughout our proofs.
- ▷ [Appendix B](#) contains proofs for generation under contamination ([Theorem 5](#) and [Informal Theorem 6](#)).
- ▷ [Appendix C](#) contains proofs for generation with density under contamination ([Theorem 7](#) and [Informal Theorem 8](#)), along with additional results for set-based and element-based density.
- ▷ [Appendix D](#) studies generation with density beyond worst-case adversaries.
- ▷ [Appendix E](#) presents additional results, including generation with unknown noise rates and generation from finite collections.
- ▷ [Appendix F](#) collects supplementary preliminaries, notation, and discussion.
- ▷ [Appendix G](#) provides further related work.

Appendix A. Outline of Algorithmic Templates

In this section we describe two algorithmic templates that we use in our proofs.

A.1. Priority-Based Intersection Meta Algorithm

Before presenting our first algorithmic template, it is instructive to first consider an algorithm due to [Charikar and Pabbaraju \(2025\)](#) for generation in the limit without contamination.

The [Charikar and Pabbaraju \(2025\)](#) Algorithm. At the n -th step, this algorithm considers the input $x_{1:n}$ and the first n languages $L_1, \dots, L_n$ of a given collection $\mathcal{L}$. We first filter for *consistent* languages, *i.e.*, languages $L_{i_n(1)}, L_{i_n(2)}, \dots$ containing all strings $x_1, \dots, x_n$ for $i_n(j) < i_n(j+1)$. Then, the algorithm computes the largest intersection in the order of indices $i_n(1), i_n(2), \dots$ such that the intersection is infinite, *i.e.*, take the intersection of the first J_n filtered languages where J_n is the largest integer such that $\left| \bigcap_{j=1}^{J_n} L_{i_n(j)} \right| = \infty$. We then output an unseen element from this intersection.

Suppose the target language L_{i^*} has index i^* and consider some L_i for $i < i^*$. Either $L_i \supseteq L_{i^*}$, or there is some string $x \in L_{i^*} \setminus L_i$ that is enumerated at some point, so that L_i is no longer consistent. Since L_{i^*} is always consistent, and the consistent languages ordered before L_{i^*} are eventually supersets, the intersection is always taken over a collection containing L_{i^*} so that the algorithm is guaranteed to generate from some infinite subset of L_{i^*} .

Our template algorithm ([Algorithm 1](#)) makes two main changes to the [Charikar and Pabbaraju \(2025\)](#) algorithm:

- (1) we change the order when taking intersections based on the priority of a language and
- (2) we introduce different stopping rules when taking the intersection.

Re-ordering by Priorities. In the presence of contamination, we cannot completely eliminate candidate languages L_i from consideration if we observe a string $x_n \notin L_i$, since it is unclear if x_n is actually a member of the target language or just noise. Instead, at step n of our algorithm,

we assign a priority $P_i^{(n)} \in \mathbb{N}$ to the language L_i . Although we state present priorities abstractly, we should think of incrementing $P_i^{(n)}$ as penalizing a candidate language L_i whenever we observe $x_n \notin L_i$. Then, we take the intersection in increasing order of priorities. Ideally, only supersets of the target language K can be ordered before K , and the intersection we compute will always be a subset of K . This recovers the (Charikar and Pabbaraju, 2025) algorithm when we take the priorities to be $P_i^{(n)} = i$ if L_i is consistent and $P_i^{(n)} = \infty$ once we observe $x_m \notin L_i$ for some $m \leq n$.

Earlier Stopping. If we would like to generate with density, we must avoid taking the intersection with too many languages, or we can end up generating from an infinite subset of the target language with density 0. We thus define an abstract stopping rule that will be instantiated differently depending on the desired guarantee.

Pseudocode. The pseudocode of our meta-algorithm is presented in Algorithm 1. Here we summarize the key notation for the reader’s convenience.

1. The priority function $P(L_i, x_{1:n})$ takes a language L_i and the current enumeration $x_{1:n}$ and outputs a priority $P_i^{(n)} \in \mathbb{N} \cup \{\infty\}$.
2. $i_n(j)$ is the index of the j -th ranked language at step n , i.e., $L_{i_n(1)}$ has the smallest priority value $P_{i_n(1)}^{(n)}$ (and thus “highest priority”) at step n and $L_{i_n(2)}$ has the second smallest priority value, etc.
3. The stopping function $J(L_{i_1}, \dots, L_{i_n}, x_{1:n})$ takes a sequence of languages L_{i_j} and the current enumeration $x_{1:n}$ and outputs a stopping index $J_n \in \mathbb{N}$.

Algorithm 1: Meta-Algorithm for Generation with Noise

Require: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; priority function

$P : \mathcal{L} \times U^* \rightarrow \mathbb{N} \cup \{\infty\}$ stopping function $J : \mathcal{L}^* \times \Omega^* \rightarrow \mathbb{N}$; enumeration $x_{1:\infty}$

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
 - 2 Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
 - 3 **for** $i = 1, 2, \dots, n$ **do**
 - 4 | Compute priority $P_i^{(n)}$ at step n for language L_i
 - 5 Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$,
i.e., for each $j \in [n - 1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$ or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and
 $i_n(j) < i_n(j + 1)$.
 - 6 Compute the stopping index J_n .
 - 7 Output $\bigcap_{j \leq J_n} L_{i_n(j)}$. // Set-based output
 - 8 Output any $w_n \in \bigcap_{j \leq J_n} L_{i_n(j)} \setminus (S_n \cup W_{n-1})$. // Element-based output
-

Analysis. One key property of the prefix-based intersection meta algorithm (Algorithm 1) is that the languages that are ranked higher than the target language will eventually stabilize, i.e., the list of such languages eventually stops changing for sufficiently large n . This is formalized in Lemma 11 below. We should think of $p \in \mathbb{N}$ below as an upper bound on the priority of the target language.

Lemma 11 (Prefix Priority Stabilization) *Suppose [Algorithm 1](#) is executed with input enumeration $x_{1:\infty}$. Let $P_i^{(n)} \in \mathbb{N} \cup \{\infty\}$ be the priority of language L_i computed in step n . Assume $P_i^{(n)}$ is non-decreasing in n and lower bounded by i , i.e., $i \leq P_i^{(n)} \leq P_i^{(n+1)}$ for all $n \geq 1$. For any $p \in \mathbb{N}$, define*

$$P_i^\infty := \lim_{n \rightarrow \infty} P_i^{(n)}, \quad \mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}.$$

Then there is a step n^* such that for all $n \geq n^*$,

- (a) $P_i^{(n)} \leq p$ for all $L_i \in \mathcal{L}(p)$
- (b) $P_i^{(n)} > p$ for all $L_i \notin \mathcal{L}(p)$
- (c) $P_i^{(n+1)} = P_i^{(n)}$ for all $L_i \in \mathcal{L}(p)$

Before proving [Lemma 11](#), we provide some intuition for the quantities involved. By the assumption that $P_i^{(n)}$ is a non-decreasing integer sequence indexed by n , we know that the limit P_i^∞ always exists. Since the sequence takes on discrete values, we can further deduce that either $P_i^{(n)} = P_i^\infty$ after some finite time n , or $P_i^\infty = \infty$ and the priority diverges. Assume for now that the priority $P_{i^*}^\infty < \infty$ of the target language L_{i^*} does not diverge. Then $\mathcal{L}(p)$ for $p := P_{i^*}^\infty$ consists of the languages whose priorities are always bounded above by the limiting priority of the target language. [Lemma 11](#) states that languages in $\mathcal{L}(p)$ will eventually be ordered before any language not in $\mathcal{L}(p)$. Thus when we take the intersection in [Algorithm 1](#) in increasing order of priorities, we will always consider languages in $\mathcal{L}(p)$ before all other languages. Note that the definition of $\mathcal{L}(p)$ does not depend on a particular step n . [Lemma 11](#) simplifies some of the analyses of our algorithms in that it suffices for us to analyze properties of $\mathcal{L}(p)$ rather than analyzing any particular stage of our algorithm.

We are now ready to prove [Lemma 11](#).

Proof [Proof of [Lemma 11](#)] First, we note that $P_i^\infty \geq P_i^{(n)} \geq i$ for any $n \geq i$. Hence $\mathcal{L}(p) \subseteq \{L_1, \dots, L_p\}$ and $P_i^{(n)} > p$ for all $n \geq i > p$. Let $n^* \geq 1$ be any integer such that for every $i \in [p]$, either

- (a) $P_i^{(n)} = P_i^\infty \leq p$ for all $n \geq n^*$, or
- (b) $P_i^{(n)} > p$ for all $n \geq n^*$.

This is guaranteed to exist since there are only finitely many limits, each of which is monotonic. The result follows. ■

Our goal will be to design a priority function for which the target language L_{i^*} has bounded priority $p = P_{i^*}^\infty < \infty$ so that $L_{i^*} \in \mathcal{L}(p)$. Recall the notation $\text{Cl}(\mathcal{L}') := \bigcap_{L \in \mathcal{L}'} L$ for a collection $\mathcal{L}'$ of languages. If the stopping rule also ensures that the prefix class $\mathcal{H}_n := \{L_{i_n(j)} : j \leq J_n\}$ eventually contains $\mathcal{L}(p)$ and $|\text{Cl}(\mathcal{H}_n)| = \infty$, then the algorithm generates from $\mathcal{L}(p) \subseteq L_{i^*}$. This observation is summarized below.

Corollary 12 (Sufficient Condition for Generation) *Suppose the assumptions of [Lemma 11](#) hold. If in addition the prefix class $\mathcal{H}_n := \{L_{i_n(j)} : j \leq J_n\}$ satisfies $\mathcal{H}_n \supseteq \mathcal{L}(p)$ and $|\text{Cl}(\mathcal{H}_n)| = \infty$ for all sufficiently large n , then [Algorithm 1](#) generates from $\text{Cl}(\mathcal{L}(p))$.*

Instantiations of Algorithm 1. In this paper, we make use of the meta algorithm Algorithm 1 to prove Theorem 5, Theorem 16, Theorem 27, Theorem 28, and Theorem 39. We now illustrate a few examples of instantiations of Algorithm 1.

For our algorithms that generate in the limit under vanishing noise (Appendix B.1) and constant noise (Appendix B.2), we instantiate the priority $P_i^{(n)}$ of language L_i at time step n so that, roughly speaking, $P_i^{(n)}$ increases every time the empirical noise rate $R(L_i; x_{1:n})$ exceeds some threshold c_i . For a vanishing noise rate, we can set c_i to be any positive sequence with small sum $\sum_i c_i < \varepsilon$. This ensures that we prioritize languages that have very small empirical error and the total error is at most an ε -fraction of the input enumeration. For a constant noise rate c , we set $c_i = c$ for all i , so that we prioritize languages whose noise rate eventually falls below c . The stopping condition in these two cases is identical to the (Charikar and Pabbaraju, 2025) algorithm.

Our algorithms for generation with density under M -bounded enumerations with vanishing noise (Appendix D) are similar to the algorithm for generation under vanishing noise with two key distinctions. In addition to incrementing the priority $P_i^{(n)}$ of language L_i at step n when the empirical noise rate exceeds some threshold, we also increment $P_i^{(n)}$ if the current input violates the M -boundedness condition with respect to L_i . This ensures we de-prioritize languages for which the input enumeration cannot be M -bounded. We also require a more involved stopping rule which ensures that the intersection we take is dense with respect to every language over which the intersection is taken over. This ensures that if the target language is included in the intersection, then we will always generate from some dense subset of the target language.

A.2. Finite Expansion Sub-Routine

Next, we present a simple but quite useful subroutine which we call finite expansion that we used in regimes with finite amount of contamination. The subroutine is as follows:

Algorithm 2: Sub-Routine for Finite Expansion

Require: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$

1 **Output** $\widetilde{\mathcal{L}} := \{L_{A,B} := L \cup A \setminus B : L \in \mathcal{L}, A \subseteq U \setminus L, B \subseteq L, |A| < \infty, |B| < \infty\}$.

We already explained why this subroutine is useful in Section 4. Here, we prove some simple results which will be useful later.

A.2.1. ILLUSTRATIVE APPLICATION 1 AND PROOF OF COROLLARY 9

To provide some intuition about this subroutine, we show that it can be used to conclude that language generation in the limit is possible for all countable collections with finite contamination. This, in particular, recovers as a result of (Raman and Raman, 2025) for finite amount of noise and also implies Corollary 9 that answers an open question in (Raman and Raman, 2025). We also use this result later to prove Theorem 7.

Let $\mathbb{G}$ be an element-based (or set-based) generator that generates from any countable collection in the limit without contamination, and let $\widetilde{\mathcal{L}}$ be the expanded collection constructed from Algorithm 2. Then applying $\mathbb{G}$ to $\widetilde{\mathcal{L}}$ achieves generation in the limit with finite contamination:

Proposition 13 *There is a generator that for all countable collections $\mathcal{L}$ generates from $\mathcal{L}$ under the limit under finite contamination.*

Proof As explained above, let $\mathbb{G}$ be an element-based or set-based generator that generates from any countable collection in the element without contamination. We will show that the proof works for either type of generator (as we require both later in the paper).

We start with element-based generator. Notice that, since $x_{1:\infty}$ contains finitely many noisy examples and omits finitely many elements from K , it corresponds to an enumeration of $K_{A,B} \in \mathcal{L}$ without noise or omissions, where A is the set of noisy elements and B is the set of omitted elements. Thus, $\mathbb{G}$ executed on $x_{1:\infty}$ and $\mathcal{L}$ generates in the limit from $K_{A,B}$. Hence, there exists some $n^* \in \mathbb{N}$ such that for all $n \geq n^*$ its output is a fresh element of $K_{A,B}$. Next, notice that since $|A| < \infty$ it must be the case that there is some n' such that for all $n \geq n'$ the output of the algorithm is a fresh element of $K_{A,B} \setminus A$. Since $(K_{A,B} \setminus A) \subseteq K$ we have shown that $\mathbb{G}$ generates K in the limit.

Similarly, if $\mathbb{G}$ is a set-based generator that generates in the limit from $K_{A,B}$, there exists some $n^* \in \mathbb{N}$ such that for all $n \geq n^*$, the output set A_n is a subset of $K_{A,B}$. Again, since $|A| < \infty$ and $x_{1:\infty}$ is an enumeration of $K_{A,B} = K \cup A \setminus B$ without noise or omissions, there exists some n' such that for all $n \geq n'$, we have $K_{A,B} \setminus \{x_1, \dots, x_n\} \subseteq K_{A,B} \setminus A \subseteq K$. Since the output set satisfies $A_n \subseteq K_{A,B} \setminus \{x_1, \dots, x_n\} \subseteq K$ for all $n \geq \max\{n^*, n'\}$, $\mathbb{G}$ generates K in the limit. ■

As an immediate corollary of the above proof, we can conclude [Corollary 9](#). The proof is as follows.

Proof [Proof of [Corollary 9](#)] We will instantiate [Algorithm 2](#) with the algorithm $\mathbb{G}_{\text{KM}}$ of [Kleinberg and Mullainathan \(2024\)](#), which given membership access to countable collection $\mathcal{L}$ generates from $\mathcal{L}$ in the limit, provided the inputs have no contamination example. Due to [Proposition 13](#), the only thing that is left to verify is that we can implement a membership oracle to $\mathcal{L}$ from a membership oracle to $\widetilde{\mathcal{L}}$. We can do this by keeping track of the way we enumerate languages in $\widetilde{\mathcal{L}}$. In other words, if $\widetilde{L}_i$ is the i -th language of $\mathcal{L}$ we can maintain a mapping to $L_{j,A,B} \in \mathcal{L}$, for some j, A, B . Then, the query “is $u_\ell \in \widetilde{L}_i$ ” is answered by checking if $u_\ell \in L_j \cup A \setminus B$; since we have membership access to L_j , it suffices to check if $u_\ell \in L_j$ and then, since we have explicit description of the (finite) sets A, B , we can check if $u_\ell \in A, u_\ell \in B$. ■

A.2.2. ILLUSTRATIVE APPLICATION 2 AND VARIANT OF FINITE EXPANSION

We will also consider a slightly different subroutine that only expands by adding a finite set to the existing languages, as opposed to the one in [Algorithm 2](#) that allows both addition of finitely many elements and deletion of finitely many elements. This lets us conclude a slightly stronger version of [Proposition 13](#).

Algorithm 3: Alternative Sub-Routine for Finite Expansion

Require: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$

1 Output $\widetilde{\mathcal{L}} := \{L_A := L \cup A : L \in \mathcal{L}, A \subseteq U \setminus L, |A| < \infty\}$.

Lemma 14 *There is a generator that for all countable collections $\mathcal{L}$ generates from $\mathcal{L}$ under the limit under finite noise and arbitrary omissions.*

The proof is almost identical to that of [Proposition 13](#): the only difference is that now use [Algorithm 3](#) instead of [Algorithm 2](#) for constructing $\widetilde{\mathcal{L}}$. We omit details.

Appendix B. Proofs for Generation under Contamination

In this section, we prove our results for language generation in the limit with contamination.

B.1. Proof of Theorem 5 (Generation with Vanishing Noise Rate and Arbitrary Omissions)

In this section, we prove Theorem 5, which we restate below.

Theorem 5 (Generation with Contamination) *There is a generator that, for any countable language collection $\mathcal{L}$, generates in the limit from $\mathcal{L}$ under $o(1)$ -noise and arbitrary omissions.*

The algorithm in the above result is a variant of generic meta-algorithm Algorithm 1. We present the algorithm in Algorithm 4 and highlight the changes from Algorithm 1 are highlighted in blue. Recall the following notation related to the priorities from our generic meta-algorithm Algorithm 1. For a fixed $p \geq 1$,

$$P_i^\infty := \lim_{n \rightarrow \infty} P_i^{(n)}, \quad \mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}, \quad \text{and} \quad \text{Cl}(\mathcal{L}(p)) := \bigcap_{L \in \mathcal{L}(p)} L.$$

To prove Theorem 5, we will use the following observation.

Lemma 15 (Sufficient Condition to Generate using Algorithm 4) *Consider executing Algorithm 4 with some arbitrary choice of thresholds c_i . Fix $p \geq 1$ and suppose that the set $\mathcal{L}(p)$ satisfies $|\text{Cl}(\mathcal{L}(p))| = \infty$. Then Algorithm 4 generates from $\text{Cl}(\mathcal{L}(p))$ in the limit.*

Proof [Proof of Lemma 15] We argue using Corollary 12. To do so, we first note that the priorities $P_i^{(n)}$ are non-decreasing by construction. Define the prefix class $\mathcal{L}_n = \{L_{i_n(j)} : j \leq J_n\}$. By the definition of Algorithm 4, this set always has infinite intersection. Thus to apply Corollary 12, it suffices to show that $\mathcal{L}(p) \subseteq \mathcal{L}_n$ for all sufficiently large n .

To see this, first note that the priorities $P_i^{(n)}$ of every L_i is non-decreasing by definition. Hence by Lemma 11, there is some $n^* \in \mathbb{N}$ such that $\mathcal{L}(p)$ is ordered before all other languages for every $n \geq n^*$. But $|\text{Cl}(\mathcal{L}(p))| = \infty$ by assumption so that $\mathcal{L}_n$ must contain $\mathcal{L}(p)$ for $n \geq n^*$ by the definition of the stopping rule in Algorithm 4. ■

We are now ready to prove Theorem 5.

Proof [Proof of Theorem 5] Similar to the case of finite noise rate, our strategy is to apply Lemma 15 to argue that Algorithm 4 generates from $\mathcal{L}$, but this time with threshold parameters $c_i := \frac{1}{2^{i+1}}$.

Write $K = L_{i^*}$. Set $p := P_{i^*}^\infty$ and note that $p < \infty$ since by the $o(1)$ -noise rate assumption, $R(L_{i^*}; x_{1:n}) \leq c_{i^*}$ for all sufficiently large n . If we show that $|\text{Cl}(\mathcal{L}(p))| = \infty$, then we can apply Lemma 15 to conclude that Algorithm 4 generates from $\text{Cl}(\mathcal{L}(p)) \supseteq K$.

By Lemma 11, since the priorities are non-decreasing, there is some $\bar{n} \in \mathbb{N}$ such that for $n \geq \bar{n}$, the priorities $P_i^{(n)}$ of languages $L_i \in \mathcal{L}(p)$ remain constant. For $n \geq \bar{n}$, we see that

$$|\text{Cl}(\mathcal{L}(p))| \geq |S_n \cap \text{Cl}(\mathcal{L}(p))| \geq n \left(1 - \sum_{L_i \in \mathcal{L}(p)} R(L_i; x_{1:n}) \right) \geq n \left(1 - \sum_{i \geq 1} \frac{1}{2^{i+1}} \right) \geq \frac{n}{2}.$$

In particular, $|\text{Cl}(\mathcal{L}(p))| = \infty$ as desired. ■

Algorithm 4: Algorithm for [Theorems 5](#) and [16](#)

Require: Countable collection $\mathcal{L}$; thresholds $c_1, c_2, \dots \in (0, 1)$; enumeration $x_{1:\infty}$

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
 - 2 Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
 - 3 **for** $i = 1, 2, \dots, n$ **do**
 - 4 Compute the the number steps the error rate of L_i exceeded c_i

$$N_i^{(n)} \leftarrow |\{m \in [n] : R(L_i; x_{1:m}) > c_i\}|.$$
 - 5 Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
 - 6 Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$,
i.e., for each $j \in [n-1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$ or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and $i_n(j) < i_n(j+1)$.
 - 7 Compute the largest index J_n such that the intersection in the re-ordering up to $L_{i_n(J_n)}$ is infinite
$$J_n \leftarrow \sup \left\{ \bar{j} \geq 1 : \left| \bigcap_{j=1}^{\bar{j}} L_{i_n(j)} \right| = \infty \right\}.$$
 - 8 Output any $w_n \in \bigcap_{j \leq J_n} L_{i_n(j)} \setminus (S_n \cup W_{n-1})$.
-

B.2. Proof of [Informal Theorem 6](#) (Generation with c -Noise and Arbitrary Omissions)

In this section, we prove [Theorem 16](#) which is the formal version of [Informal Theorem 6](#).

Theorem 16 *Fix any constant $c \in (0, 1)$. A countable collection $\mathcal{L}$ is generable with c -noise and arbitrary omissions if and only if the following c -constant noise generation property holds.*

Condition 1 (c -constant noise generation property) *For every non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and every enumeration $x_{1:\infty}$, either*

- (a) *there is some $L' \in \mathcal{L}'$ such that $R(L'; x_{1:n}) > c$ infinitely often, or*
- (b) *the intersection $|\text{Cl}(\mathcal{L}')| = \infty$ is infinite.*

Example 1 (Not Generatable under c -Noisy Enumerations) *For any $k \in \mathbb{N}$, there are simple enumerations and collections of just k languages $L_1, \dots, L_k$ that do not satisfy [Condition 1](#) for $c = 1/k$. For instance, for $i \in [k]$, we can take $L_i := \{n \in \mathbb{N} : n \bmod k \neq i\}$ to be the positive integers with remainder different from i modulo k , and $x_{1:\infty}$ to be the canonical enumeration of the integers.*

NECESSITY OF [CONDITION 1](#) (CONSTANT NOISE GENERATION PROPERTY)

We first prove the necessity of [Condition 1](#).

Proof [Proof of [Theorem 16](#) (Necessity)] Suppose that the c -constant noise generation property does not hold. Then, there is some non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and enumeration $x_{1:\infty}$ such that for every $L' \in \mathcal{L}'$, $R(L'; x_{1:n}) \leq c$ for all sufficiently large n , but the intersection $|\text{Cl}(\mathcal{L}')| < \infty$ is finite. Note this implies that $|\mathcal{L}'| \geq 2$.

Suppose towards a contradiction that $\mathcal{L}$ is generated by some generator $\mathbb{G}$. Order $\mathcal{L}' = \{L_1, \dots, L_k\}$ for some $2 \leq k < \infty$. Since $x_{1:\infty}$ is a valid enumeration with c -noise of each $L_j \in \mathcal{L}'$, $\mathbb{G}$ must simultaneously generate from all L_j for $j \in [k]$ given the same enumeration $x_{1:\infty}$. We will argue that this yields a contradiction.

Let $S_n := \{x_1, \dots, x_n\}$, $w_n := \mathbb{G}_n(x_{1:n})$, and $W_n := \{w_1, \dots, w_n\}$ denote the set of input and generated strings up to time n . By assumption, for each $j \in [k]$, there is some n_j^* such that $w_n \in L_j \setminus (S_n \cup W_{n-1})$ for all $n \geq n_j^*$. Take $n^* := \max_{j \in [k]} n_j^*$. Then

$$w_n \in \bigcap_{j \in [k]} (L_j \setminus (S_n \cup W_{n-1})) = \left(\bigcap_{j \in [k]} L_j \right) \setminus (S_n \cup W_{n-1}) \subseteq \text{Cl}(\mathcal{L}') \setminus W_{n-1}$$

for all $n \geq n^*$. However, this is impossible for $n > n^* + |\text{Cl}(\mathcal{L}')|$ as $\mathbb{G}$ has exhausted all elements in the finite intersection. ■

Next, we design an element-based generation algorithm to show that the constant noise generation property suffices to guarantee generation. Interestingly, the same algorithm achieves generation with unknown vanishing error as well as generation with arbitrary omissions.

SUFFICIENCY OF [CONDITION 1](#) (CONSTANT NOISE GENERATION PROPERTY)

Next, we prove the sufficiency of [Condition 1](#).

Pseudocode. The pseudocode is [Algorithm 4](#) instantiated with uniform thresholds $c_i = c$.

Analysis. We are now ready to prove the other direction of our characterization of constant noise generation ([Theorem 16](#)).

Proof [Proof of [Theorem 16](#) (Sufficiency)] Suppose that the c -constant noise generation property holds. We will apply [Lemma 15](#) to prove that [Algorithm 4](#) can generate when instantiated with thresholds $c_1 = c_2 = \dots = c$.

Write $K = L_{i^*}$ and define $p := P_{i^*}^\infty$. By definition, there is some n_{i^*} such that $R(L_{i^*}; x_{1:n}) \leq c$ for $n \geq n_{i^*}$. Thus for $n \geq n_{i^*}$, $P_{i^*}^{(n)} = p < \infty$ remains constant moving forwards. By [Lemma 15](#), it suffices to check that $|\text{Cl}(\mathcal{L}(p))| = \infty$ in order to ensure that [Algorithm 4](#) generates from $\text{Cl}(\mathcal{L}(p)) \supseteq K$. Now, $|\mathcal{L}(p)| \leq p$ is finite and by [Lemma 11](#), the priorities $P_i^{(n)}$ of all its members $L_i \in \mathcal{L}(p)$ stabilize after some finite time. By the definition of the priorities in [Algorithm 4](#), this means that no $L_i \in \mathcal{L}(p)$ can satisfy $R(L_i, x_{1:n}) > c$ infinitely often. But then by the c -constant noise generation property, $|\text{Cl}(\mathcal{L}(p))| = \infty$.

This concludes the proof. ■

Appendix C. Proofs for Generation with Density under Contamination

In this section, we prove our results for language generation with density under contamination. First, we prove [Theorem 7](#) in [Appendix C.1](#). Then, we prove [Informal Theorem 8](#) in [Appendix C.2](#). Finally, we present our other characterizations for dense generation mentioned in [Table 1](#).

C.1. Proof of [Theorem 7](#) (Dense Generation with Finite Contamination I)

In this section, we prove [Theorem 7](#) in three parts corresponding to the claims in [Theorem 7](#).

C.1.1. PROOF OF [THEOREM 7](#) (I) (SET-BASED UPPER DENSITY UNDER FINITE NOISE AND CONSTANT OMISSION DENSITY)

In this section, we prove claim (i) of [Theorem 7](#) via the following theorem, which establishes a stronger result.

Theorem 17 *For all countable collections $\mathcal{L}$, there is a generator that generates in the limit from $\mathcal{L}$ with set-based upper density at least $1 - c$ under finite noise and c -omissions.*

Thus, under finite noise and finite omissions, the above generator has set-based upper density 1 proving claim (i) of [Theorem 7](#). We also show [Theorem 17](#) is tight:

Theorem 18 *There is a countable collection $\mathcal{L}$, such that no generator can generate in the limit with set-based upper density at least $1 - c + \varepsilon$ for any $\varepsilon > 0$, under c -omissions without noise.*

The proof of [Theorem 18](#) appears at the end of this section.

PROOF OF [THEOREM 17](#)

To prove [Theorem 17](#), we will first prove the following result in the noiseless setting, and then make use of a finite-expansion.

Proposition 19 *For all countable collections $\mathcal{L}$, there is a generator that generates in the limit from $\mathcal{L}$ with set-based upper density at least $1 - c$ under c -omissions and no noise.*

Proof [Proof of [Proposition 19](#)] We will show that [Algorithm 5](#) works for [Proposition 19](#).

Let $K = L_{i^*} \in \mathcal{L}$ be the target language, and $x_{1:\infty}$ be an enumeration of K without noise and with c -omissions. Let $\hat{K} \subseteq K$ be the subset of K such that $x_{1:\infty}$ is an enumeration of $\hat{K}$ without noise or omissions. By assumption, $\mu_{\text{low}}(\hat{K}, K) \geq 1 - c$.

Generation Guarantee. First, we show that [Algorithm 5](#) achieves generation in the limit. We observe that for $n \geq i^* + 1$, we always have $K \in \mathcal{A}_n$. Note that if $S_n \not\subseteq L_i$, then $S_{n+1} \not\subseteq L_i$. Thus, eventually, the prefix of languages in $\mathcal{A}_n$ that come before K stabilizes, since once some language leaves the set $\mathcal{A}_n$, it will never enter it again. Let $n^* \in \mathbb{N}$ be the time such that for all $n \geq n^*$, $\mathcal{A}_n \cap \{L_i : i \leq i^*\}$ is fixed. Denote this collection as $\mathcal{A}_\infty$. Note that $K \in \mathcal{A}_\infty$. We also note that every language in $\mathcal{A}_\infty$ has to be a superset of $\hat{K}$, since each of them has to be consistent with $x_{1:\infty}$, which is an enumeration of $\hat{K}$ without noise or omissions. As a result, $\hat{K} \subseteq \text{Cl}(\mathcal{A}_\infty)$ and thus the largest common prefix $\mathcal{L}_n$ of $\mathcal{A}_{n-1}$ and $\mathcal{A}_n$ satisfies $\mathcal{A}_\infty \subseteq \mathcal{L}_n$ for $n \geq n^* + 1$. This finishes the proof of generation in the limit, as for $n \geq n^* + 1$, $\text{Cl}(\mathcal{L}_n) \subseteq \text{Cl}(\mathcal{A}_\infty) \subseteq K$.

Algorithm 5: Algorithm for [Theorem 17](#)**Require:** Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; enumeration $x_{1:\infty}$

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2 Let $\mathcal{A}_n \leftarrow \{L_i : S_n \subseteq L_i, 1 \leq i \leq n\}$ be the subcollection of languages consistent with the seen examples.
- 3 If $n = 1$, set $\mathcal{L}_n \leftarrow \emptyset$.
- 4 If $n \geq 2$, compute the largest common prefix $\mathcal{L}_n$ of $\mathcal{A}_{n-1}$ and $\mathcal{A}_n$ with infinite intersection, *i.e.*,

$$\mathcal{L}_n \leftarrow \{L_i \in \mathcal{A}_n : \{L_1, \dots, L_i\} \cap \mathcal{A}_n = \{L_1, \dots, L_i\} \cap \mathcal{A}_{n-1}, \text{ and } |\text{Cl}(\{L_1, \dots, L_i\} \cap \mathcal{A}_n)| = \infty\}.$$

- 5 Output $\text{Cl}(\mathcal{L}_n) \setminus S_n$.

Density Guarantee. Next, we show that infinitely often, [Algorithm 5](#) outputs a set with lower set density at least $1 - c$. At time n , consider $\mathcal{L}_n$. Either $\widehat{K} \subseteq \text{Cl}(\mathcal{L}_n)$, or there exists $L \in \mathcal{L}_n$ such that $\widehat{K} \not\subseteq L$. In the first case, we readily get $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n) \setminus S_n, K) \geq \mu_{\text{low}}(\widehat{K} \setminus S_n, K) \geq 1 - c$. Thus, we now assume we are in the second case.

List the languages in $\mathcal{L}_n$ in order as $\mathcal{L}_n = \{L_{i_n(1)}, \dots, L_{i_n(t)}\}$ where $t = |\mathcal{L}_n|$. For $n \geq n^* + 1$, we know that $\mathcal{A}_\infty$ appears as a nonempty prefix of $\mathcal{L}_n$, and moreover every language in $\mathcal{A}_\infty$ is a superset of $\widehat{K}$. Let L be the first language in $\mathcal{L}_n$ which is not a superset of $\widehat{K}$, which exists by the assumption. Since $L \not\subseteq \widehat{K}$ and $x_{1:\infty}$ enumerates $\widehat{K}$ without omissions, there exists some finite time $n' > n$ such that L is no longer consistent with $S_{n'}$. By our falling back strategy used in the construction of $\mathcal{L}_n$, at time n' , we set $\mathcal{L}_{n'}$ to be the prefix of languages of $\mathcal{L}_n$ coming before L . Since we assumed that L is the first language in $\mathcal{L}_n$ which is not a superset of $\widehat{K}$, we know that every language in $\mathcal{L}_{n'}$ is a superset of $\widehat{K}$, and thus $\widehat{K} \subseteq \text{Cl}(\mathcal{L}_{n'})$. This means that infinitely often, the output set $\text{Cl}(\mathcal{L}_n) \setminus S_n$ achieves lower set density at least $1 - c$ in K , concluding the proof. ■

Now we are ready to prove [Theorem 17](#) by invoking [Algorithm 3](#) and [Lemma 14](#).

Proof [Proof of [Theorem 17](#)] Consider the following generation algorithm.

The Algorithm. First, we construct the expanded collection $\widetilde{\mathcal{L}}$ via finite-expansion:

$$\widetilde{\mathcal{L}} := \{L_A := L \cup A : L \in \mathcal{L}, A \subseteq U \setminus L, |A| < \infty\}.$$

Then, we execute [Algorithm 5](#) on the collection $\widetilde{\mathcal{L}}$ and the enumeration $x_{1:\infty}$. Since $x_{1:\infty}$ is a valid enumeration of an unknown target language $K \in \mathcal{L}$, with finite noise set A and c -omission, there exists $\widehat{K} \subseteq K$ such that $x_{1:\infty}$ is an enumeration of $\widehat{K} \cup A$ without noise or omissions, and $\mu_{\text{low}}(\widehat{K}, K) \geq 1 - c$. Thus, $x_{1:\infty}$ corresponds to an enumeration for the language K_A without noise and with c -omission, where we used that $\mu_{\text{low}}(\widehat{K} \cup A, K_A) = \mu_{\text{low}}(\widehat{K}, K)$ for any finite set A .

Generation Guarantee. By [Proposition 19](#), [Algorithm 5](#) applied to $\widetilde{\mathcal{L}}$ and $x_{1:\infty}$ generates in the limit from K_A . By [Lemma 14](#), we know that the same algorithm generates in the limit from K .

Density Guarantee. If we denote the output set of [Algorithm 5](#) applied to $\widetilde{\mathcal{L}}$ and $x_{1:n}$ as A_n , by [Proposition 19](#), we know that $\limsup_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) = \limsup_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K_A) \geq 1 - c$, where we used that $K_A = K \cup A$ differs from K only by a finite set. ■

PROOF OF THEOREM 18

Proof [Proof of Theorem 18] Consider a two language collection $L_1, L_2 \subseteq U$ such that $L_1 \subseteq L_2$ and $\mu_{\text{low}}(L_1, L_2) = \mu_{\text{up}}(L_1, L_2) = 1 - c$. Consider the canonical enumeration of L_1 without noise and without omissions. Note that this enumeration is a valid enumeration for both L_1 and L_2 with c -omissions without omissions. Therefore, any set-based generator that generates in the limit from the target language K has to generate from L_1 in the limit. Since L_1 has upper density $1 - c$ in L_2 , the set-based generator achieves set-based upper density at most $1 - c$ if the adversary sets $K = L_2$. ■

C.1.2. PROOF OF THEOREM 7 (II) AND (III) (ELEMENT-BASED DENSITY WITH FINITE CONTAMINATION)

In this section, we prove claims (ii) and (iii) Theorem 7.

Theorem 20 (Claims (ii) and (iii) of Theorem 7) *For all countable collections of languages $\mathcal{L}$, there is an algorithm that achieves $1/2$ (respectively $1/8$) upper (respectively lower) element-based density under finite noise and finite omissions.*

This result follows from the following pair of results.

Theorem 21 (Element-Based Density Guarantee with Finite Contamination) *Fix $\rho \in [0, 1]$. Suppose G_{vanilla} generates any countable collection $\mathcal{L}$ in the limit with upper (respectively lower) element-based density ρ_{up} (respectively ρ_{low}) under no noise and no omissions. Then there exists a generator G_{tolerant} that achieves the same guarantees with finite contamination.*

Theorem 22 (Noiseless Element-Based Density Guarantees (Kleinberg and Wei, 2025a)) *For every countable collection of languages there is an algorithm that generates in the limit and achieves $1/2$ (respectively $1/8$) upper (respectively lower) element-based density when there is no noise or omissions.*

It remains to prove Theorem 21.

Proof [Proof of Theorem 21] The proof proceeds by reduction. We construct a new algorithm, G_{tolerant} , that uses the algorithm G_{vanilla} from Theorem 22 as a subroutine. We will show that G_{tolerant} generates from $\mathcal{L}$ under the noisy setting and inherits the density guarantees of G_{vanilla} .

Reduction. We define a new collection $\mathcal{L}'$ via finite expansion:

$$\mathcal{L}' := \{L \cup A \setminus B \mid L \in \mathcal{L}, A, B \subseteq \mathcal{X}, |A| < \infty, |B| < \infty\}.$$

Our new algorithm G_{tolerant} is defined simply as the algorithm G_{vanilla} from Theorem 22 executed on the language class $\mathcal{L}'$. The key insight is that an adversary presenting a *contaminated enumeration* of $K \in \mathcal{L}$ is indistinguishable from an adversary presenting an *enumeration of some $T \in \mathcal{L}'$ without noise or omissions*.

Generation Guarantee. From Proposition 13 implies that G_{tolerant} generates in the limit.

Density Guarantee. Let $S_{\text{out}} = (s_1, s_2, \dots)$ be the infinite output sequence from G_{tolerant} . By Theorem 22, G_{vanilla} achieves the density guarantees for its output with respect to some target $T \in \mathcal{L}'$. Formally, per the specified definition, the guarantees on the output set S_{out} are

$$\begin{aligned} \triangleright \mu_{\text{up}}(S_{\text{out}}, T) &= \limsup_{n \rightarrow \infty} \frac{|S_{\text{out}} \cap \{t_1, \dots, t_n\}|}{n} \geq \rho_{\text{up}}, \\ \triangleright \mu_{\text{low}}(S_{\text{out}}, T) &= \liminf_{n \rightarrow \infty} \frac{|S_{\text{out}} \cap \{t_1, \dots, t_n\}|}{n} \geq \rho_{\text{low}}, \end{aligned}$$

where $\{t_1, \dots, t_n\}$ is the set of the first n elements of T (based on the canonical enumeration of the domain). It is not hard to see that since the symmetric difference of K, T is finite, the exact same bounds hold when T is replaced by K . $\blacksquare$

C.2. Proof of Informal Theorem 8 (Dense Generation with Finite Contamination II)

In this section, we prove Informal Theorem 8.

Theorem 23 (Formal Statement of Informal Theorem 8) *A countable collection $\mathcal{L}$ is generable in the limit with set-based lower density $c > 0$ under finite noise and finite omissions if and only if for all $L, L' \in \mathcal{L}$ with $|L \setminus L'| < \infty$ it holds that $\mu_{\text{low}}(L, L') \geq c$.*

As a simple application of the above characterization, consider the following two language collection $\mathcal{L} = \{L_1, L_2\}$, where L_1 is the set of even numbers and L_2 is the set of all natural numbers. Then, by Theorem 23, under finite noise and finite omissions, there exists a set-based generator achieving set-based lower density $1/2$, but no generator can do better.

Even without contamination, achieving set-based lower density is challenging: Kleinberg and Wei (2025a) showed that this guarantee is achievable only if the collection avoids *infinite perfect towers*, a condition not satisfied by all countable collections. Theorem 23 is a complete characterization of when set-based lower density is achievable under finite contamination. It shows that contamination makes this significantly harder: as an immediate corollary, we show that there exist collections of just *two* languages for which no generator can achieve any non-trivial density. For comparison, in the noiseless setting, even the much stronger goal of *identification in the limit* is achievable for all finite collections.

Outline of Proof of Theorem 23. We prove Theorem 23 in two steps; first, we show that when the stated condition does not hold, then no algorithm can generate from $\mathcal{L}$ with set-based lower density; then, we show that if the condition holds, there exists an algorithm that generates from $\mathcal{L}$ with the desired lower density. The proof of Theorem 23 follows as direct corollary of these two results.

Lemma 24 *Let $\mathcal{L}$ be a countable collection of languages that contains L, L' with $|L \setminus L'| < \infty$ and $\mu_{\text{low}}(L, L') < c$. Then, no algorithm can generate from $\mathcal{L}$ in the limit with set-based lower density c under finite noise and finite omissions.*

Proof [Proof of Lemma 24] We begin by formally recalling the requirements for an algorithm G to “generate from $\mathcal{L}$ in the limit with set-based lower density c ” under the specified adversaries. Let $x = (x_1, x_2, \dots)$ be an input sequence, and let $S_n = G(x_{1:n})$ be the algorithm’s hypothesis at timestep n . Recall that x is an **enumeration** of a target language $K \in \mathcal{L}$ with finite noise and omissions if:

1. **Finite Noise:** The set $\{x_i \mid x_i \notin K\}$ is finite.
2. **Finite Omissions:** The set $K \setminus \{x_i \mid i \in \mathbb{N}\}$ is finite.

The algorithm $\mathbb{G}$ must satisfy two properties for *every* $K \in \mathcal{L}$ and *every* (contaminated) enumeration x of K :

- (i) **Generation in the Limit:** There exists an n^* such that for all $n > n^*$, it holds that $S_n \subseteq K$.
- (ii) **Density Guarantee:** $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(S_n, K) \geq c$.

We now proceed by contradiction.

Assume, for the sake of contradiction, that such an algorithm $\mathbb{G}$ exists. Let $L, L' \in \mathcal{L}$ be two languages satisfying the lemma's hypotheses:

1. $F = L \setminus L'$ is a finite set.
2. $\mu_{\text{low}}(L, L') < c$.

We construct an input x and a sequence of time indices $n_0 < n_1 < n_2 < \dots$ inductively. Let $\mu_{\text{low}}(L, L') = c - \varepsilon$ for some $\varepsilon > 0$. This construction will force $\mathbb{G}$ to fail either property (i) or (ii) in these timesteps.

The construction proceeds in phases. We treat even-numbered and odd-numbered phases differently. For any phase $k \geq 1$:

Phase $2k - 1$ (Target L'):

1. Define a potential enumeration T_{2k-1} as follows:
 - ▷ It begins with the prefix $x_{1:n_{2k-2}}$ (which is empty for $k = 1$).
 - ▷ It is followed by a complete, ordered enumeration of all elements in $L' \setminus \{x_1, \dots, x_{n_{2k-2}}\}$.
2. T_{2k-1} is an **enumeration of L' with finite noise**.
 - ▷ **Noise:** The number of elements in T_{2k-1} not in L' is bounded by $|x_{1:n_{2k-2}}| < \infty$. Thus, this noise is finite.
 - ▷ **Omissions:** By construction, T_{2k-1} contains all elements of L' , so there are 0 omissions.
3. By the definition of successful generation in this setting, $\mathbb{G}$ running on T_{2k-1} must eventually be outputting subsets of L' that have c lower density in L' . Thus, there must exist a time $n_{2k-1} > n_{2k-2}$ such that $\mu_{\text{low}}(S_{n_{2k-1}}, L') \geq c - \varepsilon/2$, and $S_{n_{2k-1}} \subseteq L'$ (where $S_{n_{2k-1}}$ is the output on the prefix $T_{2k-1}[1 : n_{2k-1}]$). Thus, since $\mu_{\text{low}}(L, L') = c - \varepsilon$ it must be the case that $S_{n_{2k-1}} \not\subseteq L$.
4. If no such n_{2k-1} exists, $\mathbb{G}$ fails the generation requirement for L' on input T_{2k-1} . We halt the construction and have found our contradiction.
5. Otherwise, set $x_{1:n_{2k-1}} = T_{2k-1}[1 : n_{2k-1}]$ and proceed.

Round $2k$ (Target L):

1. Define a text T_{2k} as follows:
 - ▷ It begins with the prefix $x_{1:n_{2k-1}}$.
 - ▷ It is followed by a complete, ordered enumeration of all elements in $L \setminus \{x_1, \dots, x_{n_{2k-1}}\}$. (This includes all elements of F not already in the prefix).

2. T_{2k} is an **enumeration of L with finite noise**.

- ▷ **Noise:** The set of elements in T_{2k} not in L is $x_{1:n_{2k-1}} \cap (L' \setminus L)$. This is a finite prefix of $L' \setminus L$, so the noise is finite.
- ▷ **Omissions:** By construction, T_{2k} contains all elements of L so there are 0 omissions.

3. By the definition of successful generation in this setting, $\mathbb{G}$ running on T_{2k} must eventually be outputting subsets of L that have $c - \varepsilon/2$ lower density in L . Thus, there must exist a time $n_{2k} > n_{2k-1}$ such that $S_{n_{2k}} \subseteq L$.
4. If no such n_{2k} exists, $\mathbb{G}$ fails the generation requirement for L on input T_{2k} . We halt and have a contradiction.
5. Otherwise, set $x_{1:n_{2k}} = T_{2k}[1 : n_{2k}]$ and proceed to the next round.

This inductive process has two possible outcomes:

Case 1: The construction halts after a finite number of rounds. As shown in step (4) of the inductive rounds, if the construction halts, it is because $\mathbb{G}$ failed to find a required output in all the rounds of this phase. This constitutes a failure of the generation requirement for either L or L' on a complete enumeration of the corresponding language with finite noise and zero omissions.

Case 2: The construction proceeds for infinitely many rounds. This process defines a single, infinite input $x = \lim_{k \rightarrow \infty} x_{1:n_k}$ and an infinite sequence of time indices $n_1 < n_2 < n_3 < \dots$.

Let us show that x is a valid enumeration of L' with finite noise and zero omissions.

- ▷ **Noise:** The set of elements in x not in L' is $x \cap (L \setminus L') = x \cap F$. Elements from F are *only* added during the “Target L ” rounds (even k). Since F is finite, x contains at most $|F|$ noisy elements. The noise is finite.
- ▷ **Omissions:** The set $L' \setminus x$ is empty. In every “Target L' ” round (odd k), the text T_{2k-1} is defined to contain *all* elements of L' . Since x is the limit of these prefixes, it must contain all of L' .

Therefore, x is a **enumeration of L' with finite noise**.

By property (ii) (Density Guarantee), A running on x must satisfy:

$$\liminf_{n \rightarrow \infty} \mu_{\text{low}}(S_n, L') \geq c.$$

However, consider the infinite subsequence of times $n_2, n_4, n_6, \dots, (n_{2k}, \dots)$. By construction, at every time n_{2k} (the end of a “Target L ” round), the algorithm’s output satisfies $S_{n_{2k}} \subseteq L$.

A key property of set-based density is that it is monotone in its first argument: if $A \subseteq B$, then $\mu_{\text{low}}(A, C) \leq \mu_{\text{low}}(B, C)$ for any C . Thus, for every $k \geq 1$:

$$S_{n_{2k}} \subseteq L \implies \mu_{\text{low}}(S_{n_{2k}}, L') \leq \mu_{\text{low}}(L, L').$$

By the lemma’s hypothesis, $\mu_{\text{low}}(L, L') < c$. This means we have an infinite subsequence of hypotheses $S_{n_{2k}}$ such that:

$$\mu_{\text{low}}(S_{n_{2k}}, L') \leq c - \varepsilon \quad \text{for all } k \geq 1.$$

This directly implies that the limit inferior of the *entire* sequence must be less than c :

$$\liminf_{n \rightarrow \infty} \mu_{\text{low}}(S_n, L') \leq \lim_{k \rightarrow \infty} \mu_{\text{low}}(S_{n_{2k}}, L') \leq \mu_{\text{low}}(L, L') < c.$$

This is a direct contradiction of the Density Guarantee (ii).

Thus, in both possible cases, the existence of algorithm $\mathbb{G}$ leads to a contradiction. In Case 1, $\mathbb{G}$ fails the Generation requirement. In Case 2, $\mathbb{G}$ fails the Density Guarantee. Therefore, no such algorithm $\mathbb{G}$ can exist. $\blacksquare$

Next, we proceed with describing an algorithm that achieves set-based lower density $c > 0$ whenever this condition holds.

Lemma 25 *Let $\mathcal{L}$ be a countable collection of languages such that for all $L, L' \in \mathcal{L}$ either $|L \setminus L'| = \infty$ or $\mu_{\text{low}}(L, L') \geq c$. Then, there exists algorithm that generates from $\mathcal{L}$ in the limit with set-based lower density c under finite noise and finite omissions.*

Proof We construct an algorithm $\mathbb{G}$ that generates from $\mathcal{L}$ in the limit and achieves the stated density guarantee, given the lemma's condition.

Algorithm $\mathbb{G}$. First, we construct $\widetilde{\mathcal{L}}$ via finite-expansion:

$$\widetilde{\mathcal{L}} := \{L_{A,B} := L \cup A \setminus B : L \in \mathcal{L}, A \subseteq U \setminus L, B \subseteq L, |A| < \infty, |B| < \infty\}.$$

Then, we execute an algorithm $\mathbb{G}_{\text{KM}}$ which obtains set-based lower density at least c , under enumerations without noise or omissions on $\widetilde{\mathcal{L}}$ and the enumeration $x_{1:\infty}$. Since $x_{1:\infty}$ is a valid enumeration of an unknown target language $K \in \mathcal{L}$, with finite noise set A and finite omission set B , this corresponds to an enumeration without noise or omissions for the language $K_{A,B}$.

The existence of $\mathbb{G}_{\text{KM}}$ follows immediately from the property of the collection (that, every subset of the target language is c -dense in the target) and the generation in the limit algorithm of (Kleinberg and Mullainathan, 2024) which is guaranteed to output subsets of K in the limit.

Generation Guarantee. Since $\mathbb{G}_{\text{KM}}$ achieves (set-based) generation in the limit with respect to $\widetilde{\mathcal{L}}$, Proposition 13 implies that it achieves (set-based) generation in the limit with respect to $\mathcal{L}$ when the enumeration contains finite amount of contamination.

Density Guarantee. We need to show that $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq c$, where A_n is the output of the algorithm during the n -th step. First, notice that since every $L, L' \in \mathcal{L}$ with $|L \setminus L'| < \infty$ satisfy $\mu_{\text{low}}(L, L') \geq c$ it must also be the case that every $L_{A,B}, L'_{A',B'} \in \widetilde{\mathcal{L}}$ with $|L_{A,B} \setminus L'_{A',B'}| < \infty$ satisfy $\mu_{\text{low}}(L_{A,B}, L'_{A',B'}) \geq c$. This is simply because if $|L_{A,B} \setminus L'_{A',B'}| < \infty$, it must be the case that $|L \setminus L'| < \infty$, and lower density is invariant to adding or subtracting finitely many elements to its arguments. Thus, guarantees of $\mathbb{G}_{\text{KM}}$ imply that $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K_{A,B}) \geq c$. Since $K, K_{A,B}$ differ on finitely many elements, this directly implies that $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq c$. $\blacksquare$

Remark 26 (Known Noise Level) *Our lower bound crucially exploits the generator's ignorance of the noise level: it only knows that noise is finite. When the generator knows a bound on the noise level, the characterization changes. This contrasts with non-dense generation, where knowledge of the noise rate does not affect which collections are generable.*

C.3. Additional Results for Set-Based Density under Contamination

In this section, we present additional results for the generation with set-based density from [Table 1](#).

C.3.1. CHARACTERIZATION OF SET-BASED LOWER & UPPER DENSITY UNDER VANISHING NOISE RATE AND ARBITRARY OMISSIONS

In this section, we characterize when set-based density is achievable under vanishing noise rate.

Theorem 27 (Characterization of Set-Based Density with Vanishing Noise Rate) *Fix a countable collection $\mathcal{L}$ and $\rho \in (0, 1]$. Under vanishing noise rate and arbitrary omissions, there exists a set-based generator that generates in the limit and achieves set-based lower density ρ if and only if the following property holds.*

Condition 2 (Vanishing Noise ρ -Dense Set-Generation) *For every non-empty finite sub-collection $\mathcal{L}' \subseteq \mathcal{L}$, if languages in $\mathcal{L}'$ share infinitely many elements, i.e., $|\text{Cl}(\mathcal{L}')| = \infty$, then for any $L \in \mathcal{L}'$, $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho$.*

As a simple application of the characterization in [Theorem 27](#), consider the following collection of languages $\mathcal{L} = \{L_1, L_2, L_3\}$, where $L_1 = \{i \in \mathbb{N} : i \not\equiv 0 \pmod{3}\}$, $L_2 = \{i \in \mathbb{N} : i \not\equiv 1 \pmod{3}\}$, $L_3 = \{i \in \mathbb{N} : i \not\equiv 2 \pmod{3}\}$. By the characterization, under vanishing noise rate and arbitrary omissions, the best set-based lower density any generator can achieve is $1/2$, since the intersection of any two languages have infinite cardinality and we can easily check the intersection has density $1/2$ in each of the two languages, whereas the intersection of all three languages is empty.

We note that [Theorem 27](#)'s characterization is sharp in the following sense:

- ▷ Whenever [Condition 2](#) does not hold, no set-based generator can generate in the limit and achieve set-based *upper density* ρ , even when the adversary is restricted to use an enumeration with $o(1)$ -noise and *without omissions*.
- ▷ Whenever [Condition 2](#) holds, there exists a set-based generator that generates in the limit and achieves set-based *lower density* ρ , even when the adversary presents an enumeration with $o(1)$ -noise and *with infinite omissions*.

In the remainder of this section we prove [Theorem 27](#).

NECESSITY OF [CONDITION 2](#)

We first prove the necessity of [Condition 2](#). As we will see, this condition is necessary even for enumerations without omission. Note that the case with enumerations can only be more challenging.

Proof [Proof of [Theorem 27](#) (Necessity)] Assume that there exists a non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ such that $|\text{Cl}(\mathcal{L}')| = \infty$, and there exists $L' \in \mathcal{L}'$ such that $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L') < \rho$.

We will construct an enumeration $x_1, x_2, x_3, \dots$ that is simultaneously an enumeration for any language $L \in \mathcal{L}'$ with $o(1)$ -noise and without omission. Given such an enumeration, any set-based generator that generates in the limit from $\mathcal{L}$ has to generate from all $L \in \mathcal{L}'$ in the limit, and thus will output a subset of $\text{Cl}(\mathcal{L}')$ for all large enough n . In particular, any subset of $\text{Cl}(\mathcal{L}')$ has lower set-density in L' strictly less than ρ . Since the adversary could have chosen $K = L'$, any set-based

generator that generates in the limit cannot even achieve set-based upper density at least ρ , even if the adversary does not omit any elements of K in its enumeration.

Now, we will construct such an enumeration $x_1, x_2, x_3, \dots$ that is simultaneously an $o(1)$ -noisy enumeration for any language $L \in \mathcal{L}'$ without omission. Define a scheduling of time $T = \{1, 2, 4, 8, \dots\}$. We will only use that $\mu_{\text{up}}(T, \mathbb{N}) = 0$. For $n \notin T$, the adversary enumerates the next element in $\text{Cl}(\mathcal{L}')$ that has not been enumerated. For $n \in T$, the adversary enumerates the next element in $(\bigcup_{L \in \mathcal{L}'} L) \setminus \text{Cl}(\mathcal{L}')$ that has not been enumerated. Clearly, this enumeration will enumerate every elements of every $L \in \mathcal{L}'$. Clearly, for any $L \in \mathcal{L}'$, if the element x_n enumerated at time n does not belong to L , we must have $n \in T$. Since T has density 0 in $\mathbb{N}$, we conclude that $x_1, x_2, x_3, \dots$ is an enumeration for any language $L \in \mathcal{L}'$ with $o(1)$ -noise and without omission. This concludes the proof. $\blacksquare$

SUFFICIENCY OF [CONDITION 2](#)

We next show that if [Condition 2](#) holds, then there exists a set-based generator that generates in the limit and achieves set-based lower density ρ , under $o(1)$ -noise and potentially infinite omissions in its enumeration.

Pseudocode. The pseudocode is presented in [Algorithm 6](#).

Algorithm 6: Algorithm for [Theorem 27](#)

Require: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; thresholds $c_1, c_2, \dots \in (0, 1)$; enumeration

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2 **for** $i = 1, 2, \dots, n$ **do**
- 3 Compute the following number

$$N_i^{(n)} \leftarrow \min \{N \geq 1 : \forall m \in \{N, \dots, n\}, R(L_i; x_{1:m}) \leq c_i\}.$$
- 4 Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
- 5 Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$, *i.e.*, for each $j \in [n - 1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$, or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and $i_n(j) < i_n(j + 1)$.
- 6 Compute the largest index $j_n \in [n]$ such that the intersection of the prefix of the sorted list of languages in the re-ordering up to $L_{i_n(j_n)}$ is infinite, *i.e.*,

$$J_n \leftarrow \max \left\{ \bar{j} \in [n] : \left| \bigcap_{j=1}^{\bar{j}} L_{i_n(j)} \right| = \infty \right\}.$$

- 7 Output $\bigcap_{j \leq J_n} L_{i_n(j)} \setminus S_n$.
-

Proof [Proof of [Theorem 27](#) (Sufficiency)]

Assume that for any non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ with $|\text{Cl}(\mathcal{L}')| = \infty$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho$ for any $L \in \mathcal{L}'$.

Now consider the set-based generator described in [Algorithm 6](#), with threshold parameters $c_i := \varepsilon/2^i$ for all $i \in \mathbb{N}$.

Note that the priority of any language $P_i^{(n)}$ is non-decreasing. Also note that the priority of the target language $K = L_{i^*}$ will remain fixed after some finite time n' , since the adversary uses an enumeration of $K = L_{i^*}$ with $o(1)$ -noise, and eventually the empirical noise rate $R(L_{i^*}; x_{1:n})$ stays below the positive threshold c_{i^*} for all large enough n . Let $p := P_{i^*}^\infty$ be the number that the priority of K stabilizes to.

By [Lemma 11](#), if we denote $\mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}$ and $\mathcal{L}^{(n)}(p) := \{L_i : P_i^{(n)} \leq p\}$, there exists n^* such that for all $n \geq n^*$, we have $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$. Moreover, we know that $K \in \mathcal{L}(p)$ by definition.

Now we will show that $|\text{Cl}(\mathcal{L}(p))| = \infty$. Consider any $n \geq \max\{n^*, p\}$, we know that $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$. For any $L_i \in \mathcal{L}^{(n)}(p)$, we have $N_i^{(n)} = P_i^{(n)} - i < p$. Thus, for all $m \in \{p, p+1, \dots, n\}$, we have $R(L_i; x_{1:m}) \leq c_i$. In particular, $R(L_i; x_{1:n}) \leq c_i$. Therefore, we get that

$$\begin{aligned} |\text{Cl}(\mathcal{L}^{(n)}(p)) \cap S_n| &= |S_n| - \left| \bigcup_{L_i \in \mathcal{L}^{(n)}} (S_n \setminus L_i) \right| \\ &\geq |S_n| - \sum_{L_i \in \mathcal{L}^{(n)}} |S_n \setminus L_i| \\ &\geq n - \sum_{L_i \in \mathcal{L}^{(n)}} n \cdot c_i \\ &\geq n(1 - \varepsilon). \end{aligned}$$

Since the above inequality holds for any $n \geq \max\{n^*, p\}$ and $\mathcal{L}(p) = \mathcal{L}^{(n)}(p)$ for $n \geq \max\{n^*, p\}$, we conclude that $|\text{Cl}(\mathcal{L}(p))| = \infty$.

Therefore, at any round time $n \geq \max\{n^*, p\}$, we know that the finite subcollection $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$ appears as a prefix of the sorted list $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$. Since the intersection of this prefix, $\text{Cl}(\mathcal{L}(p))$, has infinite cardinality, we know that the computed index $j_n \in [n]$ satisfies that $\mathcal{L}(p) \subseteq \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. Denote this subcollection as $\mathcal{L}_n := \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. By definition of j_n , $|\text{Cl}(\mathcal{L}_n)| = \infty$.

Thus, for any $n \geq \max\{n^*, p\}$, we know that $\mathcal{L}(p) = \mathcal{L}^{(n)}(p) \subseteq \mathcal{L}_n$. Moreover, $\mathcal{L}_n$ is a finite subcollection of $\mathcal{L}$, and $K \in \mathcal{L}(p) \subseteq \mathcal{L}_n$. By the [Condition 2](#), for any $L \in \mathcal{L}_n$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), L) \geq \rho$. In particular, since $K \in \mathcal{L}_n$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), K) \geq \rho$. Thus, the output

$$\bigcap_{j \leq j_n} L_{i_n(j)} \setminus S_n = \text{Cl}(\mathcal{L}_n) \setminus S_n$$

has lower density at least ρ in K , which concludes the proof. ■

C.3.2. CHARACTERIZATION OF SET-BASED LOWER & UPPER DENSITY UNDER CONSTANT NOISE RATE AND ARBITRARY OMISSIONS

In this section, we characterize when set-based density is achievable under constant noise rate and arbitrary omissions.

Theorem 28 (Characterization of Set-Based Density under Constant Noise Rate) *Fix a countable collection $\mathcal{L}$ and $c \in (0, 1]$. Under constant noise rate c and arbitrary omissions, there exists a set-based generator that generates in the limit and achieves set-based lower density ρ if and only if the following property holds.*

Condition 3 (c -constant Noise ρ -Dense Set-Generation) *For every non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and every enumeration $x_{1:\infty}$, either*

- (a) *there exists some language $L' \in \mathcal{L}'$ such that $R(L'; x_{1:n}) > c$ infinitely often, or*
- (b) *each language $L \in \mathcal{L}'$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho$.*

In fact, we will show:

- ▷ Whenever [Condition 3](#) does not hold, no set-based generator can generate in the limit and achieve set-based *upper density* ρ when the adversary presents an enumeration with c -noise arbitrary omissions.
- ▷ Whenever [Condition 3](#) holds, there exists a set-based generator that generates in the limit and achieves set-based *lower density* ρ when the adversary presents an enumeration with c -noise and arbitrary omissions.

NECESSITY OF [CONDITION 3](#)

We first prove the necessity of [Condition 3](#). The proof is similar to that of [Theorem 27](#) but simpler, since the negation of the c -constant noise rate ρ -dense set-generation property provides us with an enumeration that the adversary could use to force the density achieved by any set-based generator to be less than ρ .

Proof [Proof of [Theorem 28](#) (Necessity)] Assume that there exists a non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and an enumeration $x_1, x_2, x_3, \dots$ such that

- (a) for every $L \in \mathcal{L}'$, the $x_1, x_2, x_3, \dots$ is a c -noisy enumeration for L ,
- (b) and there exists $L' \in \mathcal{L}'$ such that $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) < \rho$.

Since $x_1, x_2, x_3, \dots$ is an enumeration with c -noise for any language in the finite subcollection $\mathcal{L}'$, any set-based generator that generates in the limit from $\mathcal{L}$ has to generate from all $L \in \mathcal{L}'$ in the limit under this enumeration. Thus, such a set-based generator will output a subset of $\text{Cl}(\mathcal{L}')$ for all large enough n . In particular, any subset of $\text{Cl}(\mathcal{L}')$ has lower set-density in L' strictly less than ρ . Since the adversary could have chosen $K = L'$, any set-based generator that generates in the limit cannot even achieve set-based upper density at least ρ . ■

SUFFICIENCY OF [CONDITION 3](#)

We next show that by slightly adapting [Algorithm 6](#) with threshold parameters $c_i := c$ and a different stopping condition, we obtain a set-based generator that achieves set-based lower density whenever [Condition 3](#) holds.

Pseudocode. The pseudocode is presented in [Algorithm 7](#).

Algorithm 7: Algorithm for [Theorem 28](#)

Require: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; $c \in (0, 1]$; enumeration $x_{1:\infty}$

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2 **for** $i = 1, 2, \dots, n$ **do**
- 3 Compute the following number

$$N_i^{(n)} \leftarrow \min \{N \geq 1 : \forall m \in \{N, \dots, n\}, R(L_i; x_{1:m}) \leq c\}.$$
- 4 Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
- 5 Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$, *i.e.*, for each $j \in [n - 1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$, or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and $i_n(j) < i_n(j + 1)$.
- 6 Compute the largest index $j_n \in [n]$ such that the intersection of the prefix of the sorted list of languages in the re-ordering up to $L_{i_n(j_n)}$ has lower set-density at least ρ for language $L_{i_n(j)}$ for any $j \leq j_n$, *i.e.*,

$$J_n \leftarrow \max \left\{ \bar{j} \in [n] : \forall j' \leq \bar{j}, \mu_{\text{low}} \left(\bigcap_{j=1}^{\bar{j}} L_{i_n(j)}, L_{i_n(j')} \right) \geq \rho \right\}.$$

- 7 Output $\bigcap_{j \leq J_n} L_{i_n(j)} \setminus S_n$.
-

Proof [Proof of [Theorem 28](#) (Sufficiency)] Assume that for any non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and every enumeration $x_1, x_2, x_3, \dots$, either

- (a) there is some $L' \in \mathcal{L}'$ such that $R(L'; x_{1:n}) > c$ infinitely often, or
- (b) each language $L \in \mathcal{L}'$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho$.

Now consider the set-based generator described in [Algorithm 6](#), with threshold parameters $c_i := c$ for all $i \in \mathbb{N}$.

Note that the priorities of any language $P_i^{(n)}$ is non-decreasing. Also note that the priority of the target language K will remain fixed after some finite time n' , since the adversary uses an enumeration with c -noise of $K = L_{i^*}$, and eventually the empirical noise rate $R(L_{i^*}; x_{1:n}) \leq c$ for all large enough n . Let $p := P_{i^*}^\infty$ be the number that the priority of K stabilizes to.

By [Lemma 11](#), if we denote $\mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}$ and $\mathcal{L}^{(n)}(p) := \{L_i : P_i^{(n)} \leq p\}$, there exists n^* such that for all $n \geq n^*$, we have $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$, and $P_i^{(n)} = P_i^\infty$ for all $L_i \in \mathcal{L}(p)$,

i.e., the priorities of all languages in $\mathcal{L}(p)$ stabilize. Moreover, we know that $K \in \mathcal{L}(p)$ by definition.

Therefore, at any round time $n \geq \max\{n^*, p\}$, we know that the finite subcollection $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$ appears as a prefix of the sorted list $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$. Moreover, since the priorities of all languages in $\mathcal{L}(p)$ stabilize, $N_i^{(n)} = P_i^{(n)} - i = P_i^\infty - i$ for any $L_i \in \mathcal{L}(p)$. In other words, by definition of $N_i^{(n)}$, the enumeration $x_1, x_2, x_3, \dots$ that the adversary uses is a c -noisy enumeration for any $L_i \in \mathcal{L}(p)$. Thus, each language $L \in \mathcal{L}(p)$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}(p)), L) \geq \rho$.

Since the intersection of this prefix, $\text{Cl}(\mathcal{L}(p))$, satisfies that $\mu_{\text{low}}(\text{Cl}(\mathcal{L}(p)), L) \geq \rho$ for any $L \in \mathcal{L}(p)$, we know that the computed index $j_n \in [n]$ satisfies that $\mathcal{L}(p) \subseteq \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. Denote this subcollection as $\mathcal{L}_n := \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. By definition of j_n , every $j' \leq j_n$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), L_{i_n(j_n)}) \geq \rho$. In particular, since $K \in \mathcal{L}(p) \subseteq \mathcal{L}_n$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), K) \geq \rho$. Thus, the output

$$\bigcap_{j \leq j_n} L_{i_n(j)} \setminus S_n = \text{Cl}(\mathcal{L}_n) \setminus S_n$$

has lower density at least ρ in K , which concludes the proof. $\blacksquare$

C.4. Additional Results for Element-Based Density under Contamination

In this section, we present additional results for the generation with element-based density from Table 1.

C.4.1. FROM SET-BASED LOWER DENSITY TO ELEMENT-BASED LOWER DENSITY

To begin with, we show that under some mild assumptions, which all the set-based generation algorithms in this paper satisfy, we may turn any generator that achieves set-based density into one that achieves element-based density.

Theorem 29 *Let $\mathcal{L}$ be a countable collection. Suppose under a specific enumeration $x_{1:\infty}$ of the target language K , potentially under contamination, there is a set-based generator $\mathbb{G}$ that generates in the limit from $\mathcal{L}$. Suppose at every round n , given $S_n = \{x_1, \dots, x_n\}$, $\mathbb{G}$ outputs an infinite set $A_n \subseteq U$ and computes a finite subcollection $\mathcal{L}_n \subseteq \mathcal{L}$ such that*

1. $K \in \mathcal{L}_n$ for all large enough n and
2. $A_n \subseteq \text{Cl}(\mathcal{L}_n) \setminus \{x_1, \dots, x_n\}$ for all n .

Assume further that $\mathbb{G}$ achieves $\mu_{\text{low}}(A_n, K) \geq \rho$, for all n sufficiently large. Then, there exists an element-based generator $\bar{\mathbb{G}}$ that, under the same enumeration $x_{1:\infty}$, generates in the limit from $\mathcal{L}$ and achieves element-based lower density at least $\rho/2$.

If we assume some set-based generator achieves set-based lower density at least ρ , then for any $\varepsilon > 0$, we have $\mu_{\text{low}}(A_n, K) \geq \rho - \varepsilon$ for all large enough n . Thus, we have the following simple corollary.

Corollary 30 *In the same setting as Theorem 29, if the set-based generator achieves set-based lower density at least ρ , i.e., $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq \rho$. Then, for any $\varepsilon > 0$, there exists an element-based generator that generates in the limit and achieves element-based lower density at least $(\rho - \varepsilon)/2$.*

We remark that the element-based generator we construct for [Theorem 29](#) assumes the knowledge of ρ .

Pseudocode. The pseudocode is presented in [Algorithm 8](#). We assume that at each round, the set-based generator computes a finite set of indices $I(n)$ such that $\mathcal{L}_n = \{L_i : i \in I(n)\}$, and outputs an infinite set $A_n \subseteq \text{Cl}(\mathcal{L}_n) \setminus \{x_1, \dots, x_n\}$. Moreover, for all sufficiently large n , $K \in \mathcal{L}_n$ and $\mu_{\text{low}}(A_n, K) \geq \rho$ for some given $\rho \in [0, 1]$.

Algorithm 8: Algorithm for [Theorem 29](#)

Require: Countable collection $\mathcal{L}$; enumeration $x_{1:\infty}$; set-based generator $\mathbb{G}$

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples enumerated by the adversary till round n
- 2 Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of elements output before round n
- 3 Compute $(A_n, \mathcal{L}_n) \leftarrow \mathbb{G}(S_n)$
- 4 Compute $m_n \in \mathbb{N}$ to be the smallest number such that for all $L \in \mathcal{L}_n$,

$$\frac{|A_n \cap \{\ell_1, \ell_2, \dots, \ell_m\}|}{m} \geq \frac{\mu_{\text{low}}(A_n, L)}{1 + 2^{-n}}, \quad \forall m \geq m_n, \quad (1)$$

where $L = \{\ell_1, \ell_2, \ell_3, \dots\}$ denote the listing of elements of L in the natural ordering.

- 5 **if** $n > 1$ **then**
- 6 Update $m_n \leftarrow \max\{m_n, m_{n-1} + 1\}$ so that the sequence $m_1 < \dots < m_n$ is increasing.
- 7 **Compute the index**

$$k(n) \leftarrow \begin{cases} \max\left\{k \in [n] : \frac{m_k}{1+2^{-k}} \leq \frac{2n}{\rho}\right\}, & \text{if } \frac{m_1}{1+2^{-1}} \leq \frac{2n}{\rho}, \\ 1, & \text{else.} \end{cases}$$

- 8 **Output the smallest** $w_n \in A_{k(n)} \setminus (S_n \cup W_{n-1})$ **in the canonical ordering.**
-

Proof [Proof of [Theorem 29](#)] If $\rho = 0$, then the statement of the theorem becomes trivial. We assume that $\rho > 0$ is a positive constant.

Fix an enumeration $x_{1:\infty}$, potentially under contamination. Let $\mathbb{G}$ be a set-based generator satisfying the assumptions in [Theorem 29](#). Consider the element-based generator $\overline{\mathbb{G}}$ described in [Algorithm 8](#).

We first show that the element-based generator $\overline{\mathbb{G}}$ in [Algorithm 8](#) is well-defined. Since $\mathcal{L}_n$ is a finite subcollection of $\mathcal{L}$, if we define $m_n(L) \in \mathbb{N}$ to be the smallest number such that (1) holds for L , then

$$m_n = \max_{L \in \mathcal{L}_n} m_n(L)$$

is well-defined. Each A_n is infinite, hence there is always some $w_n \in A_{k(n)} \setminus (S_n \cup W_{n-1})$.

Now we proceed to show that $\overline{\mathbb{G}}$ generates in the limit and achieves element-based lower density at least $\frac{\rho}{2}$.

$\overline{\mathbb{G}}$ generates in the limit: Let N_1 be sufficiently large so that $\mathcal{L}_n \ni K$ for all $n \geq N_1$. Moreover, let N_2 be sufficiently large so that $N_2 \geq N_1$ and $\frac{2n}{\rho} \geq \frac{m_{N_1}}{1+2^{-N_1}}$ for all $n \geq N_2$. Since the sequence

$\frac{m_n}{1+2^{-n}}$ is non-decreasing in n , we see that $k(n) \geq N_1$ for all $n \geq N_2$. This means that $K \in \mathcal{L}_{k(n)}$ and $A_{k(n)} \subseteq K$ for all $n \geq N_2$, so that $w_n \in A_{k(n)} \setminus (S_n \cup W_{n-1}) \subseteq K \setminus (S_n \cup W_{n-1})$ for all $n \geq N_2$.

$\overline{\mathbb{G}}$ achieves element-based lower density: Let $K = \{\ell_1, \ell_2, \dots\}$ denote the canonical ordering of K . We first prove the following claim:

Claim 31 *There exists $n^* \in \mathbb{N}$ such that $w_n \in \{\ell_1, \dots, \ell_{\lceil 2n(1+2^{-k(n)})/\rho \rceil}\}$ for all $n \geq n^*$.*

To see the claim, let N_3 be sufficiently large so that $\mathcal{L}_n \ni K$ and $\mu(A_n, K) \geq \rho$ for all $n \geq N_3$. Moreover, let N_4 be sufficiently large so that $\frac{2n}{\rho} \geq \frac{m_{N_3}}{1+2^{-N_3}}$ for all $n \geq N_4$. This means that $A_{k(n)} \subseteq K$ for all $n \geq N_4$. By construction, for all $n \geq N_4$, we have $k(n) \geq N_3$ and

$$|A_{k(n)} \cap \{\ell_1, \dots, \ell_m\}| \geq \frac{\mu_{\text{low}}(A_{k(n)}, K) \cdot m}{1 + 2^{-k(n)}} \geq \frac{\rho m}{1 + 2^{-k(n)}}$$

for all $m \geq m_{k(n)}$, and in particular for $m = \lceil 2n(1 + 2^{-k(n)})/\rho \rceil$ by construction of $k(n)$.

In other words,

$$\left| A_{k(n)} \cap \left\{ \ell_1, \dots, \ell_{\lceil 2n(1+2^{-k(n)})/\rho \rceil} \right\} \right| \geq \frac{\rho (2n(1 + 2^{-k(n)})/\rho)}{1 + 2^{-k(n)}} = 2n.$$

Since there are at most $(2n - 1)$ elements in $S_n \cup W_{n-1}$, the smallest unseen element w_n according to the canonical ordering of $A_{k(n)}$ that we output must belong to the first $\lceil 2n(1 + 2^{-k(n)})/\rho \rceil$ elements in the canonical ordering of K . This finishes the proof of the claim, in which we can take $n^* = N_4$.

With Claim 31 in hand, for $n \geq n^*$, if we define

$$t(n) := \max_{\bar{n} \leq n} \left\lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \right\rceil$$

then,

$$\frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n)}\}|}{t(n)} \geq \frac{|\{w_{n^*}, w_{n^*+1}, \dots, w_n\} \cap \{\ell_1, \dots, \ell_{t(n)}\}|}{t(n)} \geq \frac{n - n^*}{t(n)}. \quad (2)$$

Since $t(n) = \max_{\bar{n} \leq n} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil \rightarrow \infty$ as $n \rightarrow \infty$ and is non-decreasing, for all m large enough, there exists a unique number $n(m) \in \mathbb{N}$ such that

$$t(n(m)) \leq m < t(n(m) + 1).$$

Moreover, $n(m)$ is non-decreasing in m , and $n(m) \rightarrow \infty$ as $m \rightarrow \infty$. Also note that $k(n)$ is an unbounded and non-decreasing integer sequence. Since $k(n) \rightarrow \infty$ and $t(n) \rightarrow \infty$ as $n \rightarrow \infty$, we

have

$$\begin{aligned}
\mu_{\text{low}}(w_{1:\infty}, K) &= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_m\}|}{m} \\
&\geq \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m)) + 1} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \frac{t(n(m))}{t(n(m)) + 1} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \frac{\max_{\bar{n} \leq n(m)} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil}{\max_{\bar{n} \leq n(m)+1} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \frac{t(n(m))}{\max\{t(n(m)), \lceil 2(n(m) + 1)(1 + 2^{-k(n(m)+1})/\rho \rceil\}}
\end{aligned}$$

since $t(n(m)) = \max_{\bar{n} \leq n(m)} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil \geq \lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil$, we have

$$\begin{aligned}
&\geq \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \\
&\quad \cdot \frac{\lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil}{\max\{\lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil, \lceil 2(n(m) + 1)(1 + 2^{-k(n(m)+1})/\rho \rceil\}} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \min \left\{ 1, \frac{\lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil}{\lceil 2(n(m) + 1)(1 + 2^{-k(n(m)+1})/\rho \rceil} \right\}
\end{aligned}$$

since $n(m) \rightarrow \infty$ and $k(n(m)) \rightarrow \infty$ as $m \rightarrow \infty$, we get

$$= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))}$$

by inequality (2), we get

$$\geq \liminf_{m \rightarrow \infty} \frac{n(m) - n^*}{t(n(m))}$$

since $n(m)$ is non-decreasing and unbounded, we have

$$\begin{aligned}
&\geq \liminf_{n \rightarrow \infty} \frac{n - n^*}{t(n)} \\
&= \liminf_{n \rightarrow \infty} \frac{n - n^*}{\max_{\bar{n} \leq n} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil}
\end{aligned}$$

note that if $\bar{n} < n/2$, we have $2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \leq 2n/\rho \leq 2n(1 + 2^{-k(n)})/\rho$. Thus, we have

$$\begin{aligned}
&= \liminf_{n \rightarrow \infty} \frac{n - n^*}{\max_{\substack{\bar{n}: \\ n/2 \leq \bar{n} \leq n}} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil} \\
&\geq \liminf_{n \rightarrow \infty} \frac{n - n^*}{\max_{\substack{\bar{n}: \\ n/2 \leq \bar{n} \leq n}} \lceil 2n(1 + 2^{-k(\bar{n})})/\rho \rceil} \\
&= \frac{\rho}{2},
\end{aligned}$$

where the last equality follows from $k(n) \rightarrow \infty$ as $n \rightarrow \infty$. ■

C.4.2. IMPOSSIBILITY OF ELEMENT-BASED DENSITY UNDER INFINITE CONTAMINATION

Theorem 21 shows that element-based generation achieves non-trivial density under finite contamination. We now show this breaks down under infinite contamination: non-trivial element-based density is impossible, even with a vanishing noise rate and no omissions.

Theorem 32 *There exists a collection of two languages $\mathcal{L} = \{L_1, L_2\}$ and an enumeration with $o(1)$ -noise and without omissions from a target language $K \in \mathcal{L}$, such that no element-based generator can generate from K in the limit with element-based upper density at least ρ (for any $\rho > 0$).*

Proof [Proof of [Theorem 32](#)] Consider two languages L_1, L_2 with the property that $L_1 \subseteq L_2$ and $\mu_{\text{up}}(L_1, L_2) = 0$. Consider the following enumeration of the elements of L_2 .

Define a scheduling of time steps $T = \{1, 2, 4, 8, \dots\}$. We only need the fact that T has upper density 0 in $\mathbb{N}$. At time $n \in T$, the adversary outputs the next element in $L_2 \setminus L_1$ that hasn't been generated. At time $n \notin T$, the adversary outputs the next element in L_1 that hasn't been generated.

Note that the adversary will enumerate all the elements of L_2 , and thus all the elements of L_1 . Clearly, the adversary enumerates from L_2 with noise rate 0. On the other hand, since the adversary enumerates elements not belonging to L_1 only at time $n \in T$, and T has density 0 in $\mathbb{N}$, the noise rate of the adversary's enumeration is $o(1)$ in L_1 .

Since the enumeration is a valid enumeration with $o(1)$ -noise for both L_1 and L_2 , and $L_1 \subseteq L_2$, any generator generating from K in the limit has to generate from L_1 in the limit. Since L_1 has upper density 0 in L_2 , the generator achieves element-based upper density 0 if the adversary sets $K = L_2$. ■

Appendix D. Generation with Density Beyond the Worst-Case

In this section we introduce a beyond-worst-case model that restricts the order in which the adversary can present the elements of its chosen language K ; importantly, this model does not restrict the choices of the adversary in selecting the target K , it merely restricts its power to present its elements in arbitrary orders. Informally, the adversary cannot, infinitely often, enumerate elements of K that appear arbitrarily later in the canonical enumeration than the elements it has enumerated so far. In particular, under the interpretation that the canonical enumeration places “easier” elements of languages before “harder” ones, we impose the restriction that the adversary cannot, infinitely often, enumerate very hard examples before easier ones. This is inspired by practical phenomena of LLM training; indeed, folklore results known as “curriculum learning” empirically show that it is crucial that LLMs are first trained on “easier” tasks before harder ones ([Bengio et al., 2009](#); [Hacohen and Weinshall, 2019](#)).

Definition 33 (M -Bounded Displacement Enumeration) *Let $x_{1:\infty}$ be any enumeration and L be an arbitrary language. Let $\{\ell_1, \ell_2, \dots\} = L$ be the canonical enumeration of L and define*

$$\sigma(n) = \sigma(x_n, L) := \begin{cases} j, & x_n = \ell_j \in L, \\ 0, & x_n \notin L. \end{cases}$$

We say that $x_{1:\infty}$ is an M -bounded displacement enumeration with respect to L if there is some $n^* \in \mathbb{N}$ such that for all $n \geq n^*$, $\sigma(n) \leq Mn$.

Remark that by definition, $\ell_{\sigma(n)} = \ell_{\sigma(x_n, L)} = x_n$ for all $x_n \in L$. Note also that we do not require $\sigma : \mathbb{N} \rightarrow \mathbb{N} \cup \{0\}$ to be injective or surjective since it is possible that $x_{1:\infty} \neq L$ as sets.

In order to develop some intuition for [Definition 33](#), we consider several examples below. First, we consider some simple noiseless examples.

Example 2 Let $L = \mathbb{N}$ and consider the following enumerations.

- ▷ The canonical enumeration $x_n = n$ is M -bounded for $M = 1$
- ▷ Let $L := \mathbb{N} - M \cdot \mathbb{N}$ be the natural numbers that are not divisible by M . The enumeration where $x_n = M \cdot \lceil n/2 \rceil$ when n is odd and x_n being the $(n/2)$ -th element of L when n is even is $(M/2)$ -bounded.
- ▷ Let $L := \{n \in \mathbb{N} : \sqrt{n} \notin \mathbb{N}\}$ be the non-squares. The enumeration consisting of $x_n = \lceil n/2 \rceil^2$ when n is odd and x_n being the $(n/2)$ -th element of L when n is even is not M -bounded for any M .
- ▷ Finally, we note that density and M -boundedness are very different notions. For instance, even if an enumeration $x_{1:\infty}$ is dense in L , it may not be M -bounded for any M . To see this, define $x_{1:\infty}$ as follows: if i is odd, then $x_i = 2^i$, and, otherwise if i is even then $x_i = i/2$. $x_{1:\infty}$ has $\mu_{\text{low}}(x_{1:\infty}, K) = 1/2$ but is not M -bounded for any finite M .

Next, we consider a slightly more involved example with constant noise rate. As we will see in [Appendix D.1](#), [Example 3](#) demonstrates that generation under constant noise remains hard even with bounded displacement enumerations.

Example 3 Let $k \geq 2$ be an integer and define the language

$$L_i := [k] \cup (\mathbb{N} \setminus i \cdot \mathbb{N}) = \{1, \dots, k\} \cup \{n \in \mathbb{N} : n \text{ is not divisible by } i\}.$$

Consider the finite collection $\mathcal{L} := \{L_i : i \in [k]\}$ and the canonical enumeration $x_n = n$ of the natural numbers. By construction, the noise rate w.r.t. each $L_i, i \in [k]$ is bounded above by $1/k$. Moreover, we have $\sigma(x_n, L_i) \leq (1 + 1/k)n$ for all $n \geq 1, i \in [k]$. Hence $x_{1:\infty}$ is a M -bounded displacement enumeration of every $L_i, i \in [k]$ with noise rate $1/k$.

Remark 34 Intuitively, [Definition 33](#) restricts the “speed” of the enumeration to a linear speed-up compared to the canonical enumeration. Slightly more precisely, the adversary cannot place elements from deep in the tail of the canonical enumeration at the front of the input enumeration. Moreover, the adversary cannot omit too many elements from the prefix of the canonical enumeration, as otherwise, it is forced to output elements from deeper in the tail of the canonical enumeration.

It is illustrative to consider the lower bound construction for generation with density under $o(1)$ -noise enumerations from [Theorem 32](#) which consists of two languages $L' \subseteq L$. Let $L = \{\ell_1, \ell_2, \dots\}$ be the canonical enumeration of L . [Theorem 32](#) essentially chooses $L' = \{\ell_{i_1}, \ell_{i_2}, \dots\}$ where $i_n = \omega(n)$ so that L' is a sparse subsequence of L . Then, the adversary mainly enumerates from L' , occasionally outputting from L so that the enumeration is a valid $o(1)$ -noise enumeration of

both L' , L . Roughly speaking, under this enumeration, any algorithm cannot distinguish whether the target language is L or L' and is forced to generate from the sparse subset L' . However, this worst case enumeration cannot be bounded w.r.t. L since $i_n = \omega(n)$. If we are promised that the enumeration is M -bounded w.r.t. the target language, the adversary cannot maliciously enumerate from such a sparse subset of the target language.

D.1. Properties of Bounded Displacement Adversaries

In order to develop a better understanding of [Definition 33](#), we study several elementary properties.

Closed under Subsets. An immediate but useful feature of bounded displacement enumerations is that they are closed under language subsets.

Proposition 35 *Suppose the enumeration $x_{1:\infty}$ is an M -bounded displacement enumeration with respect to some language L . Then for any $L' \subseteq L$, $x_{1:\infty}$ is an M -bounded displacement enumeration w.r.t. L' .*

Proof Let $\{\ell_1, \ell_2, \dots\} = L$ be the canonical enumeration of L . The canonical enumeration of L' can be obtained from that of L by deleting some elements and decreasing the indices of remaining elements. In other words, there is some subsequence $i_1 \leq i_2 \leq \dots$ such that $\{\ell_{i_1}, \ell_{i_2}, \dots\} = L'$ is the canonical enumeration of L' . For any $x = \ell_{i_j} \in L'$, its index $j = \sigma(x, L')$ in L' is at most its index $i_j = \sigma(x, L)$ in L . Hence for all sufficiently large n ,

$$\sigma(x_n, L') \leq \sigma(x_n, L) \leq Mn,$$

as desired. ■

Change of Density. The next lemma is the key lemma for generation with lower density. Intuitively, it says that we can estimate the density of a language up to a multiplicative factor by simply computing the empirical “density” w.r.t. the *input enumeration*. This is key since we do not know the canonical enumeration of the target language (otherwise generation with density is trivial) but we do have access to the input enumeration.

Lemma 36 (Change of Density) *Let $x_{1:\infty}$ be an M -bounded displacement enumeration of a language K with arbitrary noise rate (and possibly omissions). Then for any other language $L \subseteq K$,*

$$\mu_{\text{low}}(L, K) \geq \frac{1}{M} \cdot \liminf_n \frac{|L \cap x_{1:n}|}{n} \quad \text{and} \quad \mu_{\text{up}}(L, K) \geq \frac{1}{M} \cdot \limsup_n \frac{|L \cap x_{1:n}|}{n}.$$

Proof By [Proposition 35](#), $x_{1:\infty}$ is also an M -bounded displacement enumeration w.r.t. L . By [Definition 33](#), for each $n \geq n^*$, we have $\sigma(x_n, L) \leq Mn$. Define $B := \max\{\sigma(x_n, L) : n < n^*\}$ and consider the canonical enumeration $\{\ell_1, \ell_2, \dots\}$ of K . For $n \geq \max(n^*, B)$, we have $L \cap \{x_1, \dots, x_n\} \subseteq L \cap \{\ell_1, \dots, \ell_{Mn}\}$, where we understand ℓ_0 to be a special string not in K . This implies that

$$\frac{|L \cap \{x_1, \dots, x_n\}|}{n} \leq \frac{|L \cap \{\ell_1, \dots, \ell_{Mn}\}|}{n} = M \cdot \frac{|L \cap \{\ell_1, \dots, \ell_{Mn}\}|}{Mn}.$$

Note that the limit inferior (superior) of the RHS is exactly the lower (upper) density of L in K . Thus taking the limit inferior (superior) on both sides concludes the proof. ■

Hardness of Identification. Since we have introduced a new model that is limiting the power of the adversary, it is worth understanding how restrictive this change is. Towards that end, we study how the landscape of identification changes under our definition. Our result below shows that, even for simple collections of languages, identification remains intractable for any $M > 1$. We leave a full characterization of identification for M -bounded adversaries as an interesting open question.

Theorem 37 *There is a countable collection $\mathcal{L}$ for which no algorithm can identify in the limit, even when restricted to M -bounded enumerations for any $M > 1$.*

Proof Let $\mathcal{L} = \{\mathbb{N}\} \cup \{\mathbb{N} \setminus \{n\} : n \in \mathbb{N}\}$ be the collection consisting of the natural numbers and all its subsets obtained by deleting a single number. Let $\mathcal{A}$ be any algorithm.

The adversary begins by enumerating the natural numbers $1, 2, \dots$. If $\mathcal{A}$ never outputs $\mathbb{N}$, we are done. Let n_1 be the first step when $\mathcal{A}$ outputs $\mathbb{N}$. Then at step $n_1 + 1$, we output $n_1 + 2$ instead of $n_1 + 1$ and then continue enumerating the natural numbers in the canonical order. If $\mathcal{A}$ never outputs $\mathbb{N} \setminus \{n_1 + 1\}$, we are done. Otherwise, let $n_2 > n_1$ be the first step when $\mathcal{A}$ outputs $\mathbb{N} \setminus \{n_1 + 1\}$. At step $n_2 + 1$, the adversary “fills in” the previously skipped element by outputting $n_1 + 1$, instead of $n_2 + 1$, and then continues enumerating the natural numbers in the canonical order again. Repeat.

All in all, the adversary can always ensure that $\mathcal{A}$ does not correctly identify after any finite time. On the other hand, if we let $\sigma(n)$ denote the value of the element the adversary enumerates at step n , then $\sigma(n) \leq n + 1$. Thus identification in the limit remains impossible in general even under M -bounded displacement enumerations, for any $M > 1$. ■

Hardness of Generation under Constant Noise. To further understand the landscape of generation under bounded displacement enumerations, and as earlier promised, we show that generation with finite noise remains hard.

Theorem 38 *For any $k \geq 2$, there is a finite collection of k languages $\mathcal{L}$ for which no algorithm can generate in the limit from $\mathcal{L}$ under constant $1/k$ -rate noise, even when restricted to $1/k$ -bounded enumerations.*

Proof Consider the finite family $\mathcal{L}$ from [Example 3](#). Let $k \geq 2$ be an integer and define the language $L_i := [k] \cup (\mathbb{N} \setminus i \cdot \mathbb{N}) = \{1, \dots, k\} \cup \{n \in \mathbb{N} : n \text{ is not divisible by } i\}$. Then $\mathcal{L} = \{L_i : i \in [k]\}$.

As we have seen in [Example 3](#), the canonical enumeration of the natural numbers is $1/k$ -bounded and has $1/k$ -noise w.r.t. every $L_i, i \in [k]$. Thus any hypothetical algorithm that generates from $\mathcal{L}$ under the assumed conditions must simultaneously generate from all L_i . In other words, it generates from $\bigcap_{i \in [k]} L_i = [k]$. This must be a contradiction as this is a finite set. ■

All in all, we have seen that bounded displacement enumerations does not trivialize the identification problem, nor generation under constant rate noise. [Appendix D.2](#) shows that we do gain an advantage for the regime of $o(1)$ -noise in terms of generation with density.

D.2. Results for Set-Based Density

We now present our results for generation with set-based density guarantee under bounded displacement adversaries. In [Appendix D.2.1](#), we first design an algorithm which roughly obtains $1/M$ set-based lower density under M -bounded enumerations. [Appendix D.2.2](#) demonstrates that this guarantee is tight in the worst case sense: In general, no algorithm can do better than $1/M$ in this setting.

D.2.1. GENERATION WITH SET-BASED LOWER DENSITY

Having described the beyond-worst-case model we consider, we show how these types of enumerations help achieve non-trivial lower density (in a set-based sense) under vanishing noise rate for *all* countable collections. At each step, our algorithm outputs a finite set of languages indexed by $I(n) \subseteq \mathbb{N}$ such that for sufficiently large n , their intersection is an infinite subset of the target, *i.e.*, $\cap_{i \in I(n)} L_i \subseteq K$ (generation in the limit), and is dense in K .

Theorem 39 *There is an algorithm $\mathbb{G}$ that, for any collection $\mathcal{L}$, target language $K \in \mathcal{L}$, and error parameter $\varepsilon \in (0, 1)$, given an M -bounded displacement enumeration $x_{1:\infty}$ with $o(1)$ -noise and arbitrary omissions, $\mathbb{G}$ outputs an intersection $\cap_{i \in I(n)} L_i$ of finitely many languages $|I(n)| < \infty$ with infinite cardinality $|\cap_{i \in I(n)} L_i| = \infty$ at step n such that*

- (a) $\cap_{i \in I(n)} L_i \subseteq K$ for all sufficiently large n and
- (b) $\mu_{\text{low}}(\cap_{i \in I(n)} L_i, K) \geq \frac{1-\varepsilon}{M}$ for all sufficiently large n .

Pseudocode. The pseudocode for our algorithm is an instantiation of our generic meta-algorithm [Algorithm 1](#) and is presented in [Algorithm 9](#). We highlight the changes in blue. We remark that the algorithm requires two non-standard parameters as input. $\varepsilon \in (0, 1)$ is an error parameter which is required to compute priorities appropriately. The parameter $M \geq 1$ is the boundedness parameter ([Definition 33](#)), which is required to correctly compute both the priorities and the stopping rule. [Algorithm 9](#) ensures $\frac{1-\varepsilon}{M}$ lower density.

Recall the following notation for a string x and language L with canonical enumeration $L = \{\ell_1, \ell_2, \dots\}$

$$\sigma(x, L) := \begin{cases} j, & x = \ell_j \in L, \\ 0, & x \notin L. \end{cases}$$

In other words, $\sigma(x, L)$ is the index of x in the canonical enumeration of L if $x \in L$ and otherwise $\sigma(x; L) = 0$ if $x \notin L$. As a shorthand for an enumeration $x_{1:\infty}$, we write

$$\sigma(x_{1:n}, L) := \max_{j \in [n]} \sigma(x_j, L).$$

We remark that by [Definition 33](#), any M -bounded enumeration $x_{1:\infty}$ of L satisfies $\sigma(x_{1:n}; L) \leq Mn$ for all sufficiently large n .

Analysis. Recall the following notation for a given $p \geq 1$

$$P_i^\infty = \lim_{n \rightarrow \infty} P_i^{(n)}, \quad \mathcal{L}(p) = \{L_i : P_i^\infty \leq p\}, \quad \text{Cl}(\mathcal{L}(p)) = \cap_{L \in \mathcal{L}(p)} L.$$

Algorithm 9: Algorithm for [Theorem 39](#)

Require: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; density error parameter $\varepsilon \in (0, 1)$;
enumeration $x_{1:\infty}$; bounded displacement parameter $M \geq 1$

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2 Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
- 3 **for** $i = 1, 2, \dots, n$ **do**
- 4 Compute the number of time steps m where the error rate of L_i exceeds $2^{-i}\varepsilon$ or the input enumeration $x_{1:m}$ was not M -bounded

$$N_i^{(n)} \leftarrow |m \in [n] : R_m(L_i; x_{1:m}) > 2^{-i}\varepsilon \vee \sigma(x_{1:m}; L_i) > Mm|.$$
- 5 Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
- 6 Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$,
i.e., for each $j \in [n]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$ or
 $(P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}) \wedge (i_n(j) < i_n(j+1))$.
- 7 Compute the largest index J_n such that the intersection in the re-ordering up to $L_{i_n(J_n)}$ is dense
in $L_{i_n(j)}$ for all $j \leq J_n$

$$J_n \leftarrow \max \left\{ \bar{j} \in [n] : \forall j \leq \bar{j}, \mu_{\text{low}} \left(\bigcap_{j \leq J_n} L_{i_j(n)}, L_{i_j(n)} \right) \geq \frac{1-\varepsilon}{M} \right\}.$$

- 8 Output $\bigcap_{j \leq J_n} L_{i_j(n)}$.

Proof [Proof of [Theorem 39](#)] Let $\mathcal{H}_n := \{L_{i_n(j)} : j \leq J_n\}$ denote the prefix class. Write $K = L_{i^*}$ for the target language and define $p := P_{i^*}^\infty < \infty$, which is guaranteed to exist by the assumption that $x_{1:\infty}$ is a valid M -bounded $o(1)$ -noisy enumeration of L_{i^*} . We have $L_{i^*} \in \mathcal{L}(p)$ by construction. We will argue that $\mathcal{L}(p) \subseteq \mathcal{H}_n$ for all sufficiently large n so that the output $\mathcal{L}(\mathcal{H}_n)$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{H}_n), K) \geq \frac{1-\varepsilon}{M}$ by the definition of [Algorithm 9](#) since $K \in \mathcal{L}(p) \subseteq \mathcal{H}_n$.

We first note that the priority of each language is increasing in n by definition. Hence by [Lemma 11](#), there is some $n^* \in \mathbb{N}$ such that for $n \geq n^*$, the priority $P_i^{(n)} = P_i^\infty$ of $L_i \in \mathcal{L}(p)$ no longer changes and moreover, $P_i^{(n)}$ takes on strictly smaller value than any language not in $\mathcal{L}(p)$. In other words, $\mathcal{L}(p)$ is ordered before all other languages. Thus, in order to argue that the stopping rule J_n stops after considering all members of $\mathcal{L}(p)$, it suffices to show that for any subset $\mathcal{L}' \subseteq \mathcal{L}(p)$, $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \frac{1-\varepsilon}{M}$ for every $L \in \mathcal{L}'(p)$.

Now, by the definition of the priorities in [Algorithm 9](#) for $n \geq n^*$ every $L \in \mathcal{L}' \subseteq \mathcal{L}(p)$ satisfies

$$R(L; x_{1:n}) := \frac{|x_{1:n} \setminus L|}{n} \leq \frac{\varepsilon}{2^i}.$$

Thus for $n \geq n^*$, and any non-empty $\mathcal{L}' \subseteq \mathcal{L}(p)$, we have

$$\frac{|\text{Cl}(\mathcal{L}') \cap x_{1:n}|}{n} \geq 1 - \sum_{i: L_i \in \mathcal{L}'} \frac{|x_{1:n} \setminus L_i|}{n} \geq 1 - \sum_{i \geq 1} \frac{\varepsilon}{2^i} \geq 1 - \varepsilon.$$

Also by the definition of the priorities in [Algorithm 9](#), for $n \geq n^*$, every $L \in \mathcal{L}' \subseteq \mathcal{L}(p)$ satisfies

$$\sigma(x_{1:n}, L) \leq Mn.$$

In other words, the input enumeration $x_{1:\infty}$ is also an M -bounded enumeration w.r.t. L . Thus, the change in density formula ([Lemma 36](#)) applies to the languages $\text{Cl}(\mathcal{L}') \subseteq L$ and we conclude that for all $L \in \mathcal{L}'$,

$$\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \frac{1}{M} \liminf_n \frac{|\text{Cl}(\mathcal{L}') \cap x_{1:n}|}{n} \geq \frac{1 - \varepsilon}{M}.$$

This concludes the proof by our initial remark. ■

Implications to Generation without Noise. Interestingly, [Theorem 39](#) has direct implications for generation without noise, that could be of interest beyond the scope of our work. In particular, since this result shows that (even with noise) if we restrict the adversary to enumerations with M -bounded displacements, we can achieve $1/M$ lower density in a set-based manner for *all* countable collections. Recall that ([Kleinberg and Wei, 2025a](#)) showed that one can achieve c set-based lower density under worst-case enumerations if and only if $\mathcal{L}$ does not contain an infinite perfect tower, parametrized by c . Our result shows that this lower bound can be circumvented by restricting the adversary to $1/c$ bounded enumerations. We leave a complete characterization of the landscape of this beyond-worst-case generation without noise as an interesting open direction.

D.2.2. LOWER BOUND FOR SET-BASED DENSITY

In the previous sections, we demonstrated that given an M -bounded enumeration, it is always possible to perform set-based generation with set-based lower density $\frac{1-\varepsilon}{M}$, where $\varepsilon \in (0, 1)$ is arbitrary. A natural question is whether this is the best possible. In this section, we show that in the worst case, no generator can obtain density greater than $\frac{1}{M}$, even *upper density*.

Theorem 40 *There exists a collection of two languages L_1, L_2 such that if the adversary provides an M -bounded displacement enumeration with noise rate $o(1)$ from this collection, then any generator that generates from the target K in the limit achieves set-based upper density at most $1/M$.*

The proof is via a similar construction as [Theorem 32](#), except we must construct the enumeration and collection slightly more carefully to satisfy M -boundedness.

Proof Consider two languages $L_1 = M \cdot \mathbb{N} := \{M \cdot n : n \in \mathbb{N}\}$ and $L_2 = \mathbb{N}$ with the property that $L_1 \subseteq L_2$ and $\mu_{\text{up}}(L_1, L_2) = \frac{1}{M}$. Consider the following enumeration of the elements of L_2 .

Define a scheduling of time steps $T = \{1, 2, 4, 8, \dots\}$. We only need the fact that T has density 0 in $\mathbb{N}$. Let $x_{1:\infty}$ be the following enumeration: At time $n \in T$, the adversary outputs the next element in $L_2 \setminus L_1$ that it has not output. At time $n \notin T$, the adversary outputs the next element in L_1 that it has not output. Now, $\sigma(x_n, L_1) \leq n$ since we can interpret $x_{1:\infty}$ as the canonical

enumeration of $L_1 = M \cdot \mathbb{N}$ with some elements of $L_2 \setminus L_1$ interleaved. Note that the latter can only decrease $\sigma(x_n, L_1)$. On the other hand

$$\sigma(x_n, L_2) \leq \begin{cases} n, & n \in T, \\ M \cdot n, & n \notin T. \end{cases}$$

Hence $x_{1:\infty}$ is an M -bounded enumeration w.r.t. both L_1, L_2 .

Note that the adversary will enumerate all the elements of L_2 , and thus all the elements of L_1 . Clearly, the adversary enumerates from L_2 with noise rate 0. On the other hand, since the adversary enumerates elements not belonging to L_1 only at time $n \in T$, and T has density 0 in $\mathbb{N}$, the noise rate of the adversary's enumeration is also $o(1)$ in L_1 .

Since the enumeration is a valid enumeration with $o(1)$ -noise for both L_1 and L_2 , and $L_1 \subsetneq L_2$, any set-based generator generating from K in the limit has to output $S_n \subseteq L_1$ for all sufficiently large n . Since L_1 has upper density $1/M$ in L_2 , the generator achieves set-based upper density at most $1/M$ if the adversary sets $K = L_2$. $\blacksquare$

D.3. Results for Element-Based Density

In this section, we extend our results for set-based density to element-based density. [Appendix D.3.1](#) applies our black-box reduction from set-based generation with density to roughly attain element-based density $\frac{1}{2M}$ with noise under M -bounded displacement enumerations. Similarly, we show that no algorithm can do better than $\frac{1}{2M}$ in general within [Appendix D.3.2](#). Note that, unlike the set-based density guarantee, our upper and lower bounds have a gap of $\frac{1}{2M}$.

D.3.1. GENERATION WITH ELEMENT-BASED LOWER DENSITY

In this section, we design an algorithm for element-based generation with lower density given an M -bounded enumeration.

Theorem 41 *There is an generator $\mathbb{G}$ that, for any collection $\mathcal{L}$, target language $K \in \mathcal{L}$, and error parameter $\varepsilon \in (0, 1)$, given an M -bounded displacement enumeration $x_{1:\infty}$ with $o(1)$ -noise rate and arbitrary omissions, the output $w_{1:\infty}$ of $\mathbb{G}$ has lower density*

$$\mu_{\text{low}}(w_{1:\infty}, K) \geq \frac{1 - \varepsilon}{2M}.$$

Having completed the heavy lifting of set-based generation with density in [Appendix D.2.1](#), we can now leverage our black-box reduction from set-based generation with density to element-based generation with density from [Appendix C.4.1](#) to design a simple algorithm.

Proof Our [Algorithm 9](#) achieving lower set-based density ([Theorem 39](#)) falls under the set-based generation algorithm in the premise of [Theorem 29](#). Hence, we can apply the black-box reduction from [Theorem 29](#) to construct the desired generator. $\blacksquare$

D.3.2. LOWER BOUND FOR ELEMENT-BASED UPPER DENSITY

We now demonstrate that the same counterexample for set-based density also extends to the element-based density under bounded displacement enumerations.

Theorem 42 *There exists a collection of two languages L_1, L_2 such that if the adversary provides an M -bounded displacement enumeration with $o(1)$ -noise from this collection, then any generator that generates from the target K in the limit achieves element-based upper density at most $1/M$.*

The proof is identical to that of [Theorem 40](#). In the setting where arbitrary enumeration is allowed, the adversary can choose to skip enumerating elements that the generator outputs and force the generator to obtain density at most $1/2$. We can picture this as skipping every other element of some given enumeration. However, if the original enumeration is M -bounded, skipping every other element leads to a $2M$ -bounded enumeration. Thus, the adversary's strategy for arbitrary enumerations does not hold for bounded enumerations. This is the main reason we have a gap of $\frac{1}{2M}$ between the density guarantee of [Theorem 41](#) and the lower bound in [Theorem 42](#).

Proof Consider two languages $L_1 = M \cdot \mathbb{N} := \{M \cdot n : n \in \mathbb{N}\}$ and $L_2 = \mathbb{N}$ with the property that $L_1 \subseteq L_2$ and $\mu_{\text{up}}(L_1, L_2) = \frac{1}{M}$. Consider the following enumeration of the elements of L_2 .

Define a scheduling of time steps $T = \{1, 2, 4, 8, \dots\}$. We only need the fact that T has density 0 in $\mathbb{N}$. Let $x_{1:\infty}$ be the following enumeration: At time $n \in T$, the adversary outputs the next element in $L_2 \setminus L_1$ that it has not output. At time $n \notin T$, the adversary outputs the next element in L_1 that it has not output. Now, $\sigma(x_n, L_1) \leq n$ since we can interpret $x_{1:\infty}$ as the canonical enumeration of $L_1 = M \cdot \mathbb{N}$ with some elements of $L_2 \setminus L_1$ interleaved. Note that the latter can only decrease $\sigma(x_n, L_1)$. On the other hand

$$\sigma(x_n, L_2) \leq \begin{cases} n, & n \in T, \\ M \cdot n, & n \notin T. \end{cases}$$

Hence $x_{1:\infty}$ is an M -bounded enumeration w.r.t. both L_1, L_2 .

Note that the adversary will enumerate all the elements of L_2 , and thus all the elements of L_1 . Clearly, the adversary enumerates from L_2 with noise rate 0. On the other hand, since the adversary enumerates elements not belonging to L_1 only at time $n \in T$, and T has density 0 in $\mathbb{N}$, the noise rate of the adversary's enumeration is also $o(1)$ in L_1 .

Since the enumeration is a valid $o(1)$ -noisy enumeration for both L_1 and L_2 , and $L_1 \subsetneq L_2$, any set-based generator generating from K in the limit has to output $x_n \in L_1$ for all sufficiently large n . Since L_1 has upper density $1/M$ in L_2 , the generator achieves set-based upper density at most $1/M$ if the adversary sets $K = L_2$. $\blacksquare$

We leave as an open question the tight density guarantee given an M -bounded enumeration.

Appendix E. Additional Results for Generation with Contamination

E.1. Generation with Vanishing and Unknown Noise Rate: Sorting by Index Only

Fix a countable collection $\mathcal{L}$. Set $c_i := \frac{1}{2^{i+1}}$. At time n , let $\mathcal{L}^{(n)} \subseteq \mathcal{L}$ be the subcollection of languages L_i with empirical noise rate $R(L_i; x_{1:n}) \leq c_i$.

Proposition 43 *Fix a subset of languages $\mathcal{L}_S := \{L_i : i \in S\}$. If $|\text{Cl}(\mathcal{L}_S)| < \infty$, then there exists T such that for $n \geq T$, $\mathcal{L}_S \not\subseteq \mathcal{L}^{(n)}$, i.e., there exists at least one language $L_i \in \mathcal{L}_S \setminus \mathcal{L}^{(n)}$.*

Proof Assume that $\mathcal{L}_S \subseteq \mathcal{L}^{(n)}$. Then, we have

$$\begin{aligned} |\text{Cl}(\mathcal{L}_S) \cap \{x_1, \dots, x_n\}| &= |\{x_1, \dots, x_n\}| - \left| \bigcup_{i \in S} (\{x_1, \dots, x_n\} \setminus L_i) \right| \\ &\geq n - \sum_{i \in S} |\{x_1, \dots, x_n\} \setminus L_i| \\ &\geq n - \sum_{i \in S} n \cdot c_i \end{aligned}$$

since $L_i \in \mathcal{L}_S \subseteq \mathcal{L}^{(n)}$,

$$\geq \frac{n}{2},$$

where the last inequality follows from $\sum_{i=1}^{\infty} c_i \leq \frac{1}{2}$.

Since $|\text{Cl}(\mathcal{L}_S)| < \infty$, we know that the above inequality stops holding if $n > 2|\text{Cl}(\mathcal{L}_S)|$. Thus, there is a finite time T so that for $n \geq T$, $\mathcal{L}_S \not\subseteq \mathcal{L}^{(n)}$. $\blacksquare$

Now we prove the following algorithm also generates in the limit under vanishing noise rate, where we replaced the priority in [Algorithm 4](#) with simply sorting by indices of the languages with low empirical noise rate. Note that in particular, this algorithm does not enjoy the stable prefix property guaranteed by [Lemma 11](#).

Algorithm 10: Algorithm for [Theorem 44](#)

Require: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; thresholds $c_1, c_2, \dots \in (0, 1)$; enumeration

$x_{1:\infty}$

- 1 Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2 Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
- 3 Let $\mathcal{A}_n \leftarrow \{L_i : R(L_i; x_{1:n}) \leq c_i, 1 \leq i \leq n\}$ be the active set of languages with empirical noise rate bounded by c_i
- 4 Compute the longest prefix $\mathcal{L}_n$ of $\mathcal{A}_n$ with infinite intersection, i.e.,

$$\mathcal{L}_n \leftarrow \sup \{L_i \in \mathcal{A}_n : |\text{Cl}(\{L_1, \dots, L_i\} \cap \mathcal{A}_n)| = \infty\}.$$

- 5 Output any $w_n \in \text{Cl}(\mathcal{L}_n) \setminus (S_n \cup W_{n-1})$.
-

Theorem 44 Let $\mathcal{L}$ be a countable collection of languages. Then, [Algorithm 10](#) generates in the limit from $\mathcal{L}$ under $o(1)$ -noise and infinite omissions.

Proof Recall that $\mathcal{L}^{(n)} := \{L_i : R(L_i; x_{1:n}) \leq c_i\}$. Since the adversary enumerates from K with vanishing noise rate, we know there exists a time T' such that for all $n \geq T'$, $K \in \mathcal{L}^{(n)}$ and $K \in \{L_1, \dots, L_n\}$, and thus $K \in \mathcal{A}_n$.

Suppose $K = L_{i^*}$. Now consider the set

$$B = \left\{ S \subseteq [i^*] : \left| \bigcap_{i \in S} L_i \right| < \infty \right\}.$$

For any $S \in B$, by [Proposition 43](#), there exists T_S such that for $n \geq T_S$, $\{L_i : i \in S\} \not\subseteq \mathcal{L}^{(n)}$. Without loss of generality, we may assume $T_S \geq T'$ for any $S \in B$.

Let $T = \max_{S \in B} T_S$. For $n \geq T$, we must have $\{i : L_i \in \mathcal{L}^{(n)}\} \cap [i^*] \notin B$, since for $n \geq T$, $\{L_i : i \in S\} \not\subseteq \mathcal{L}^{(n)}$ for any $S \in B$. Therefore, for $n \geq \max\{T, i^*\}$, $K = L_{i^*} \in \mathcal{L}^{(n)}$, and the intersection of the prefix up to K , $\mathcal{L}^{(n)} \cap \{L_1, \dots, L_{i^*}\}$, has infinite number of elements. This means that the generator will generate $w_n \in \text{Cl}(\mathcal{L}_n)$ from the intersection of a prefix $\mathcal{L}_n$ of $\mathcal{A}_n \supseteq \mathcal{L}^{(n)} \cap \{L_1, \dots, L_{i^*}\}$ containing $K = L_{i^*}$, and thus generating in the limit. ■

E.2. Generating from Finite Collections with Noise

In [Theorem 16](#), we characterize the collections that can be generated at noise rate c . In this section, we ask the dual question. We consider a class of collections, namely all finite collections of size at most k , and ask what is the highest noise rate $c > 0$ under which all these collections can be generated in the limit. We prove the following result.

Theorem 45 *For each $k \geq 2$, the following holds:*

1. *If $c \geq 1/k$, then there is a finite collection $\mathcal{L}$ of size k that not generable in the limit under c -noise.*
2. *If $c < 1/k$, then all finite collections $\mathcal{L}$ of size k can be generated in the limit under c -noise.*

NECESSARY CONDITION FOR GENERATION FROM FINITE COLLECTIONS

First, we prove the necessary condition.

Proof [Proof of [Theorem 45](#) (Necessity)] Let $\mathcal{X} = \mathbb{N}$ and, for $i \in \{0, 1, \dots, k-1\}$, define

$$L_{i+1} = \mathbb{N} \setminus \{n \in \mathbb{N} : n \equiv i \pmod{k}\}.$$

In words, L_{i+1} omits exactly the residue class $i \pmod{k}$.

Adversary. Consider the adversary's enumeration $x_t = t$ (the canonical ordering). Note that this is a valid ordering for all languages in $\mathcal{L}$ with the fraction of noisy examples being $1/k + o(1)$; therefore the noise density is $c = 1/k$ in the limit.

Impossibility of Generation. Fix any generator producing outputs $w_1, w_2, \dots$ with $w_t \notin \{x_1, \dots, x_t\}$ for all $t \geq t^*$ for some finite t^* . Among the k residue classes modulo k , one class $r \pmod{k}$ appears infinitely often in the output sequence $\{w_t\}_{t \geq 1}$ by the pigeonhole principle. Take the target language to be $K = L_{r+1}$ (which is possible due to the earlier observation that the adversaries stream is valid for languages in $\mathcal{L}$). Then, whenever $w_t \equiv r \pmod{k}$, we have $w_t \notin K$ by construction; hence the generator makes infinitely many mistakes and, thus, fails to generate in the limit from K . ■

SUFFICIENT CONDITION FOR GENERATION FROM FINITE COLLECTIONS

Next, we prove the sufficiency part of [Theorem 45](#).

Overview of the algorithm and proof. At a high level, our algorithm is as follows: at each time t , we compute a robust version space allowing up to n_t mistakes among the first t distinct examples, with n_t carefully chosen so that it grows linearly in t and slightly faster than the realized number of noisy examples. The latter condition ensures that the robust version space eventually contains the true language K and a more careful argument shows that all languages in the version space share infinitely many elements.

Before formalizing our result we begin with some preliminaries.

Preliminaries: Bounds on the Noisy Closure Dimension. Let $\mathcal{L} = \{L_1, \dots, L_k\}$ be a finite class (where each L_i infinite) and let $K \in \mathcal{L}$ be the target language. For a finite sequence $x_{1:d}$ and a noise-budget $n \geq 0$, define

$$V(x_{1:d}; n) = \{L \in \mathcal{L} : |\{x_{1:d}\} \cap L| \geq d - n\}, \quad \text{and} \quad \langle x_{1:d} \rangle_{\mathcal{L}, n} = \bigcap_{L \in V(x_{1:d}; n)} L$$

(with $\langle x_{1:d} \rangle_{\mathcal{L}, n} = \perp$ if $V(x_{1:d}; n) = \emptyset$). The n -Noisy Closure dimension $\text{NC}_n(\mathcal{L})$ is the largest d for which there exist distinct $x_{1:d}$ with $\langle x_{1:d} \rangle_{\mathcal{L}, n} \neq \perp$ but finite. For a finite size- k collection $\mathcal{L}$, [\(Raman and Raman, 2025\)](#) show that there is a class-dependent constant $d^* < \infty$ such that⁷

$$\text{NC}_n(\mathcal{L}) \leq nk + d^* + 1 \quad \text{for all } n \in \mathbb{N}. \quad (3)$$

(See their Corollary 3.4 in [\(Raman and Raman, 2025\)](#) and its proof for the construction of d^* .)

Our Result. Next, we present the main result of this section

Theorem 46 (Finite classes tolerate density $\nu < 1/k$) Fix any size- k collection $\mathcal{L} = \{L_1, \dots, L_k\}$ and $\nu < 1/k$. There exists a generator $\mathbb{G}$ (not knowing ν) that generates from K in the limit against every adversary whose noise rate is at most ν in the limit.

Proof Recall that we consider a density-bounded adversary that presents a *density-bounded noisy enumeration* of K , i.e., a stream of distinct examples $(x_t)_{t \geq 1}$ in which every $x \in K$ appears at some finite time and $\limsup_{d \rightarrow \infty} \frac{1}{d} |\{t \leq d : x_t \notin K\}| \leq \nu$. Let $S_t = \{x_1, \dots, x_t\}$ be the stream of examples presented in the first t steps and $d_t = |S_t|$ be the size of S_t . Define the dynamic noise budget

$$n_t = \max \left\{ 0, \left\lfloor \frac{d_t - (d^* + 2)}{k} \right\rfloor \right\}. \quad (4)$$

Generation $\mathbb{G}$. We claim that it is sufficient for $\mathbb{G}$ proceed as follows: At time t ,

1. Compute n_t by (4) and the robust noisy-closure $C_t := \langle x_{1:t} \rangle_{\mathcal{L}, n_t}$.
2. If $C_t \neq \perp$, output any $w_t \in C_t \setminus S_t$. Otherwise, output any arbitrary placeholder string.

Correctness of $\mathbb{G}$. In the remainder of the proof, we prove $\mathbb{G}$'s correctness in several steps.

7. The constant d^* depends only on $\mathcal{L}$ (finite) and can be precomputed from it as in the proof of Corollary 3.4 of [\(Raman and Raman, 2025\)](#).

Claim 47 (Sufficiency of Dynamic budget) *There is a finite $T < \infty$, s.t., for all $t \geq T$, $N_t \leq n_t$.*

Proof [Proof of Claim 47] Let $N_t := |\{i \leq t : x_i \notin K\}|$ be the number of corruptions by time t . Due to the limsup condition, there exists an $\varepsilon \in (0, (1/k) - c)$ and t_0 large enough that for all $t \geq t_0$,

$$N_t \leq (c + \varepsilon) d_t. \quad (5)$$

On the other hand, by (4), for t with d_t sufficiently large we have

$$n_t \geq \frac{d_t - (d^* + 2)}{k} - 1 = \frac{d_t}{k} - \frac{d^* + 2 + k}{k}.$$

Combining with (5) and using $c + \varepsilon < \frac{1}{k}$, we obtain $N_t \leq n_t$ for all $t \geq T$ for some finite T . In particular, it is sufficient to select $T = t_0 + \frac{1}{\varepsilon k}(d^* + 2 + k)$. ■

Claim 48 (Closure is Eventually Infinite) *For any $t \geq T$ (for T from Claim 47), $|C_t| = \infty$ and $C_t \subseteq K$.*

Proof Fix any $t \geq T$. Since $N_t \leq n_t$, the true language K belongs to $V(x_{1:t}; n_t)$ and, hence, $C_t = \langle x_{1:t} \rangle_{\mathcal{L}, n_t} \neq \perp$. Moreover, because $d_t > \text{NC}_{n_t}(\mathcal{L})$ by (3) and the definition of n_t , the noisy-closure C_t is either empty or infinite (by the definition of the noisy closure dimension); moreover, since the previous sentence rules out empty, C_t must be infinite. Finally, $C_t \subseteq K$ because $K \in V(x_{1:t}; n_t)$ and C_t is the intersection of supports over $V(x_{1:t}; n_t)$. ■

Now we are ready to conclude: By Claim 48, for all $t \geq T$ the set $C_t \setminus S_t$ is nonempty and contained in K . Thus, G outputs $w_t \in K \setminus S_t$ for all $t \geq T$. ■

Remark 49 (On the criticality of $c = 1/k$) *The sufficiency proof reveals why the constant $1/k$ appears. The generator succeeds eventually when its noise budget n_t satisfies both*

$$c < \frac{n_t}{d_t} \quad \text{and} \quad \text{NC}_{n_t} < d_t$$

for all $t \geq T$ for some finite T . The first condition ensures K enters the version space, while the second (with K in the version space) ensures the intersection of all version space languages is infinite. Since the noisy closure dimension $\text{NC}_n(\cdot)$ scales as $\Theta(nk)$ (see Equation (3)), the $1/k$ bound emerges from the above constraints:

- ▷ *The first condition requires $n_t = \Omega(cd_t)$.*
- ▷ *The second condition, using $\text{NC}_{n_t} = \Theta(n_t k)$, requires $\Omega(cd_t k) \leq \text{NC}_{n_t} < d_t$.*

Thus $cd_t k < d_t$, implying $c < 1/k$.

Appendix F. Additional Preliminaries and Discussion

F.1. Refresher on Limit Inferior and Limit Superior

In this section, we present the definitions of $\limsup$ and $\liminf$ that are used throughout the paper.

Definition 50 *The limit inferior and limit superior of a sequence of reals $a = (a_n)_{n \in \mathbb{N}}$ are defined as:*

$$\liminf_{n \rightarrow \infty} a_n := \sup_{N \in \mathbb{N}} \inf_{n \geq N} a_n \quad \text{and} \quad \limsup_{n \rightarrow \infty} a_n := \inf_{N \in \mathbb{N}} \sup_{n \geq N} a_n.$$

Note that, unlike $\lim_{n \rightarrow \infty} a_n$, limit inferior and limit superior exist for all sequences of reals a . We note that these limits are allowed to take values in the extended reals $\mathbb{R} \cup \{\pm\infty\}$. Some simple facts are as follows, we refer the reader to (Rudin, 1976) for more extensive discussion.

Fact 51 *Given a sequence of real numbers $a = (a_n)_{n \in \mathbb{N}}$, the following always hold:*

1. $\liminf a_n \leq \limsup a_n$
2. *If $\liminf a_n = \limsup a_n$, the $\lim_{n \rightarrow \infty} a_n$ exists and $\liminf a_n = \lim_{n \rightarrow \infty} a_n = \limsup a_n$.*

Next, we discuss some examples to build intuition.

Example 4 *The sequence $a_n = (-1)^n$ has $\liminf a_n = -1$ and $\limsup a_n = 1$*

Example 5 *Consider the sequence $a = (a_n)_{n \in \mathbb{N}}$ where $a_n = 1$ if $n = 2^m$ for some $m \in \mathbb{N}$ and, otherwise, $a_n = 0$. It has $\liminf a_n = 0$ and $\limsup a_n = 1$.*

Example 6 *The sequence $a_n = (-1)^n + 1/n$ has $\liminf a_n = -1$ and $\limsup a_n = 1$. In contrast, the related sequence $b_n = 1 + (-1)^n/n$ satisfies $\liminf b_n = \limsup b_n = 1$.*

F.2. Additional Notation

Next, we introduce some additional and standard notation.

Universe and Enumerations. Let Σ be a finite alphabet. We use Σ^* to denote the set of all finite strings over Σ . We often call Σ^* the *universe* and use U to denote it. We will often take $U = \Sigma^*$ to be the set of natural numbers $\mathbb{N}$. This is without loss of generality as Σ^* is a countable set because Σ is finite. Given a sequence $x = (x_1, x_2, x_3, \dots)$ of elements of U , we use $x_{1:n}$ to denote the prefix of elements $x_1, x_2, \dots, x_n$.

Closure and Canonical Enumeration. For us, a *language* is any infinite subset $L \subseteq U$. Given A countable collection $\mathcal{L}$ of languages (i.e., a collection of infinite sets in U), the closure of $\mathcal{L}$ is the common intersection of all languages in $\mathcal{L}$ and is denoted as $\text{Cl}(\mathcal{L}) := \bigcap_{L \in \mathcal{L}} L$. When talking about densities, we will also endow the universe U with a canonical ordering $u_1, u_2, \dots$. When $U = \mathbb{N}$, the canonical enumeration will always be $1, 2, 3, \dots$. This canonical ordering induces a canonical ordering of each language $L \subseteq U$ and, for language L , we denote it by $\ell_1, \ell_2, \dots$.

Remark 52 (Infinite Cardinality of Languages) *Following (Kleinberg and Mullainathan, 2024) and other prior works, we will always assume that each language $L \in \mathcal{L}$ has infinite cardinality, as otherwise, when $K = L$, the set of unseen valid elements a generator can output, namely, $K \setminus S_n$, can become empty.*

F.3. Examples of Lower and Upper Densities

To give some intuition about $\mu_{\text{up}}(\cdot)$ and $\mu_{\text{low}}(\cdot)$ (see [Definition 2](#)), consider the following sets

$$E = 2\mathbb{N} := \{2n\}_{n \in \mathbb{N}}, \quad P = \{n \text{ is prime}\}_{n \in \mathbb{N}}, \quad \text{and} \quad F = \cup_{i \in \mathbb{N}} [(2i)!, (2i+1)!].$$

These languages have the following densities in the universe $U = \mathbb{N}$:

$$\mu_{\text{up}}(E, U) = \mu_{\text{low}}(E, U) = \frac{1}{2}, \quad \mu_{\text{up}}(P, U) = \mu_{\text{low}}(P, U) = 0, \quad \mu_{\text{up}}(F, U) = 1 \quad \text{and} \quad \mu_{\text{low}}(F, U) = 0.$$

Hence, as we can see, lower and upper densities of L in $\mathbb{N}$ can be different for L and the lower and upper density for one language in $\mathbb{N}$ can be the same or different. Due to the definition of limit inferior and limit superior, it always holds that $\mu_{\text{low}}(L, U) \leq \mu_{\text{up}}(L, U)$.

F.4. Separation of Proper and Improper Learning with Noise or Omissions

Index-based generation can be thought as “properly” learning to generate from language K , while both element-based and set-based generations correspond to “improper” learning as they do not require the generator’s outputs to be languages in $\mathcal{L}$.

Our first example shows that index-based generation is too restrictive even for a two-language collection when the adversary’s enumeration contains 1 noisy example.

Example 7 Take $U := \mathbb{N} = \{1, 2, 3, \dots\}$, $L_1 := \mathbb{N} \setminus \{1\}$, $L_2 := \mathbb{N} \setminus \{2\}$, and $\mathcal{L} = \{L_1, L_2\}$. If the adversary chooses a complete enumeration of $\mathbb{N}$, then clearly this enumeration is an enumeration with 1 noisy example for both L_1 and L_2 with one noisy example and thus the adversary is free to declare the target language $K = L_1$ or $K = L_2$. However, outputting any index $i_n \in \{1, 2\}$ does not guarantee $L_{i_n} \subseteq K$.

Hence, we mainly focus on element-based and set-based generation in the noisy setting. Our next result shows that even for simple collections consisting of two languages, and very restricted adversaries that omit only a single element from K , index-based generation is provably impossible.

Example 8 Consider the same example as in [Example 7](#): $L_1 := \mathbb{N} \setminus \{1\}$, $L_2 := \mathbb{N} \setminus \{2\}$, and $\mathcal{L} = \{L_1, L_2\}$. If the adversary enumerates $3, 4, 5, \dots$, this enumeration is a valid enumeration with 1 element omitted for both L_1 and L_2 . However, neither index $i_n \in \{1, 2\}$ guarantees $L_{i_n} \subseteq K$.

F.5. What Type of Access Does the Generator Have To $\mathcal{L}$?

In this section, we clarify what the learner can do and cannot do.

What the learner cannot do: The learner cannot query the unknown target language K . It also cannot query future adversary behavior: at round n , only the elements $\{x_1, \dots, x_n\}$ enumerated so far are visible.

What the learner can do: The learner may query the language collection $\mathcal{L}$ via oracles, as long as queries do not involve K . Standard oracles from prior work include:

- ▷ (Membership Oracle) Given index i and element $w \in U$, return $\mathbb{1}\{w \in L_i\}$.
- ▷ (Subset Oracle) Given indices i, j , return $\mathbb{1}\{L_i \subseteq L_j\}$.

▷ (Intersection Oracle) Given a finite index set I , return $\mathbb{1}\{|\cap_{i \in I} L_i| = \infty\}$.

We additionally use:

▷ (Density Oracle) Given language L and a set A of the form $(\cap_{i \in I} L_i) \setminus F$ for finite I and F , return $\mu_{low}(A, L)$.

▷ (Density Rate Oracle) Given language L , set $A = (\cap_{i \in I} L_i) \setminus F$, and $\varepsilon > 0$, return the smallest m^* such that $\frac{|A \cap \{\kappa_1, \dots, \kappa_m\}|}{m} \geq \frac{\mu_{low}(A, L)}{1+\varepsilon}$ for all $m \geq m^*$, where $\kappa_1, \kappa_2, \dots$ is the canonical enumeration of L .

We specify which oracles each generator requires.

F.6. Generation with Repetitions in the Input Enumeration

We first remark that our hardness results hold when allowing for repetitions as we have proven them for the special case with 0 repetition.

We briefly sketch how to handle repetitions in the input enumeration within our algorithms. For an enumeration $x_1, x_2, \dots$, possibly with repetitions, we define the sub-enumeration $x_{i_1}, x_{i_2}, \dots$ consisting of the first occurrence of the unique elements in $x_{1:\infty}$. That is, $x_{i_1:\infty}$ is obtained from $x_{1:\infty}$ by deleting elements x_n that already occurred at some step $m < n$. Modify all our assumptions so that they hold with respect to the unique elements observed so far. For example, redefine the empirical noise rate so that the denominator is the number of unique elements seen so far $R(x_{1:n}, L) := \frac{|x_{1:n} \setminus L|}{|x_{1:n}|}$.

Set-Based Generation. Hypothetically, if we could execute our algorithms on $x_{i_1:\infty}$, we would be done. This is essentially the case for set-based generation, as running our algorithm on $x_{1:\infty}$ is equivalent to running our algorithm on $x_{i_1:\infty}$, except we can simply output the same set on time steps when the adversary enumerates a repeated element. Note that this suffices for both generation and generation with density.

Element-Based Generation. The case of element-based generation is slightly more involved. Fortunately, our element-based algorithms (with the exception of our reduction to (Kleinberg and Wei, 2025a), which already handles repetitions), first compute an infinite set as a function of only the adversary's enumeration, and then output an unseen element from this set. Thus we can also perform the same reduction as set-based generation, with the exception that on time steps when the adversary enumerates a repeated element, we output the first unseen element from the set chosen in the last step when the adversary did not enumerate a repeat.

It is easy to see that this algorithm generates in the limit as long as the original algorithm generates in the limit with no repetitions. To see that density guarantees still hold, we remark that the set of elements generated on input $x_{1:n}$ is a superset of the elements generated on $x_{i_1:n}$. Indeed, in step n when the adversary enumerates a unique element, the algorithm will have identified the same set as the hypothetical algorithm executed on the unique sub-enumeration. It will also output the same first unseen element, unless it has already been output due to a previous step caused by a repetition. Note here we crucially used the fact that the infinite set we are picking from is only a function of the (unique) input enumeration.

F.7. Discussion of [Bai et al. \(2026\)](#)’s Approach

The approach of ([Bai et al., 2026](#)) for generation with omissions is fragile to noise. They show that if a generator for $\mathcal{L}$ begins (valid) generation after n^* iterations in the noise-free, omission-free setting (where n^* depends on $K \in \mathcal{L}$ but not on the adversary’s enumeration of K) then the same generator handles arbitrary omissions. Such a generator for countable collections was given by ([Charikar and Pabbaraju, 2025](#)). However, this approach fails with noise. To see this, consider the collection $\mathcal{L} = \{K, L_1, L_2, \dots\}$ where

$$\begin{aligned} K &= \mathbb{N}, \\ L_i &= \{-1, -2\} \cup \{1, 2, \dots, i-1\} \cup T, \quad \text{for each } i \in \mathbb{N}, \\ T &= \{\dots, -102, -101, -100\}. \end{aligned}$$

Let the target be $K = \mathbb{N}$ with enumeration $E = (-2, -1, 1, 2, 3, \dots)$, containing just two noisy examples. At iteration $n \geq 2$, after observing $\{-1, -2, 1, \dots, n-2\}$, the only consistent language among $L_1, \dots, L_n$ is L_{n-1} . The algorithm of ([Charikar and Pabbaraju, 2025](#)) therefore generates from $T \subset L_{n-1}$. Since $T \cap K = \emptyset$, it never generates valid elements. This demonstrates that noise-free generation strategies cannot be directly applied to noisy settings, even with minimal noise.

F.8. Comparison to [Kleinberg and Wei \(2025b\)](#)

Recently, [Kleinberg and Wei \(2025b\)](#) independently and concurrently followed up on their earlier work ([Kleinberg and Wei, 2025a](#)). The follow-up ([Kleinberg and Wei, 2025b](#)) explores dense language generation under omissions (but without noise). They prove the following results:

- R1. Their main result is a generator that, for any countable collection $\mathcal{L}$, given an enumeration with c -omissions (see [Definition 4](#) for definition), generates $\mathcal{L}$ in the limit with element-based lower density at least $(1-c)/2$ (which is also tight).
- R2. A key ingredient in this result is a generator that, for any countable collection $\mathcal{L}$, given enumeration $(x_1, x_2, \dots)$, infinitely many times t outputs a set G_t with $\{x_1, x_2, \dots\} \cap G_t \subseteq K$.
- R3. Apart from this, they (1) study a notion of partial identification in the limit, providing a characterization, and (2) observe that index-based or *proper* learning is too restrictive in the presence of omissions.

We also prove R2 above using a very similar algorithm to theirs ([Theorem 17](#)), as well as observe that proper learning is impossible with omission ([Example 8](#)). These are the only explicitly shared results between our works. Adding to R3, we show that proper learning is impossible with noise ([Example 7](#)). ([Kleinberg and Wei, 2025b](#)) takes the generator in R2 to further prove R1. We take the generator in R2 further in a different direction to handle various amounts of noise (see [Table 1](#)); and, given its repeated use, we formalize this as an algorithmic template capturing it which could have broader use in the study of language generation ([Appendix A.1](#)). Apart from this, we also study the beyond-worst case model and prove stronger dense-generation guarantees under this model ([Appendix D](#)).

Appendix G. Further Related Work on Language Generation

In [Section 5](#), we presented works on language generation in the limit most directly related to our contributions. Here, we discuss additional works on language generation that provide useful context but are less central to our results.

Language Generation with Uncountable Collections. As we mentioned in [Section 1](#), ([Kleinberg and Mullainathan, 2024](#)) introduced the model of language generation in the limit. While they and many subsequent works, including ours, focus on the setting where the collection of languages is countable, [Li et al. \(2025\)](#) extended the study to uncountable collections of languages. ([Li et al., 2025](#)) introduced two more fine-grained notions of language generation: uniform and non-uniform generation. Uniform generation requires the number of strings n^* the generator needs to see before starting to generate (see [Definition 1](#)) to be independent of the choice of the target language K . Non-uniform generation allows n^* to depend on K , but requires it to be independent of the enumeration of K chosen by the adversary. (Note that generation in the limit allows n^* to be dependent of both K and the enumeration of K .) ([Li et al., 2025](#)) characterized the collections that are uniformly generable as well as collections that are non-uniformly generable. The latter characterization, in particular, demonstrates that all countable collections can be non-uniformly generated although with a different algorithm than that of ([Kleinberg and Mullainathan, 2024](#)); this result was also concurrently and independently obtained by [Charikar and Pabbaraju \(2025\)](#). Interestingly, [Charikar and Pabbaraju \(2025\)](#) showed that this type of generation is *impossible* if the learner only has membership oracle access to $\mathcal{L}$. ([Li et al., 2025](#)) gave several sufficient conditions for uncountable collections to be generable in the limit and left a complete characterization as an open problem. En route to obtaining a complete characterization, they left other open problems, notably checking closedness under finite unions: if $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k$ are generable in the limit, then is $\cup_{i=1}^k \mathcal{L}_i$ also generable in the limit? ([Hanneke et al., 2025](#); [Bai et al., 2026](#)) concurrently and independently resolved this problem by showing that the statement is false even with $k = 2$. Their proofs are near-identical and rely on a diagonalization argument which has also been featured in other works on language generation as discussed in [Section 5](#).

Further Works on Language Generation. Apart from the aforementioned works on language generation, there are two other recent papers studying this framework. [Charikar and Pabbaraju \(2026\)](#) study a notion of Pareto-optimal non-uniform generation, which seeks to obtain algorithms that achieve the Pareto frontier for non-uniform generation. On a different front, [Karbasi et al. \(2025\)](#) propose and study a theoretical model of hallucination detection that is inspired by the language identification and generation settings. Lastly, [Vafa et al. \(2024\)](#) propose a theoretical model that measures the proximity between the language represented by an LLM and the language it was trained on.