Pareto-optimal Non-uniform Language Generation

Moses Charikar MOSES@CS.STANFORD.EDU
Stanford University

Chirag Pabbaraju CPABBARA@CS.STANFORD.EDU
Stanford University

Editors: Matus Telgarsky and Jonathan Ullman

Abstract

Kleinberg and Mullainathan (2024) recently proposed an interesting model for language generation in the limit: Given a countable collection of languages, and an adversary enumerating the strings of some language L from the collection, the objective is to generate *new* strings from the target language, such that all strings generated beyond some finite time are valid. Li, Raman, and Tewari (2024) and Charikar and Pabbaraju (2024) showed strong *non-uniform* generation guarantees in this model, giving algorithms that generate new valid strings from L after seeing a number of distinct input strings $t(L)$ that depends only on L (and the collection), but *not* the enumeration order. However, for both these works, the language-wise *generation times* $t(L)$ of the algorithm can be strictly sub-optimal.

In this work, we study *Pareto-optimality* of non-uniform language generation in the limit. We propose an algorithm, whose generation times $t^\star(L)$ are (almost) Pareto-optimal: any other algorithm whose generation time for some language L is strictly smaller than $t^\star(L)$, *must satisfy* that its generation time for some *other* language L' is strictly worse than $t^\star(L')$. Pareto-optimality is essentially the best that one can achieve for non-uniform generation. Our algorithmic framework conveniently adapts to further give Pareto-optimal non-uniform generation algorithms in the practically motivated settings of *noisy* as well as *representative* generation.

Keywords: Language Generation in the Limit, Non-uniform Language Generation, Pareto-optimality.

1. Introduction

The phenomenal success of large language models in generating coherent language motivates the study of concrete theoretical models to explain their working. Synthesizing a formal model that is tractable to analyze, but also representative enough, is challenging, given the extremely complex architecture of these large language models. Nevertheless, the foundational work of Gold (1967) and Angluin (1979, 1980) on language identification provides an excellent starting point, and motivated the recent formulation of language generation in the limit by Kleinberg and Mullainathan (2024). The setup is as follows: there is a countable collection $\mathcal{C} = (L_1, L_2, L_3, \ldots)$ of languages (where each language is an infinite subset of strings from a common universe). An adversary chooses a target language L, and lists the strings in L in an online fashion according to an ordering of their choice. This corresponds to the "training data" that is input to a generative language model which receives a string as input and produces a string as output at every time step. The goal of the learning algorithm is *generation in the limit*, i.e., to guarantee that beyond some finite time, the algorithm *only* generates new strings from L that have not yet appeared in the input. Under this model, Kleinberg and Mullainathan (2024) establish a surprisingly general positive result: there exists an algorithm that achieves the guarantee of generation in the limit for *any* countable collection $\mathcal{C}$.

However, the time at which the algorithm correctly starts generating can be very sensitive to the precise order in which the language is being enumerated as input; as such, the adversary might be able to arbitrarily delay the time of correct generation, which may be undesirable. This motivates the notion of *non-uniform* language generation introduced in the work of Li et al. (2024). Non-uniform language generation requires that for every language L that the adversary may chose from the collection, the algorithm starts generating validly as soon as the input comprises of $t(L)$ many distinct strings, *independent* of what these strings are, and what order they are enumerated in. This is a considerably stronger guarantee compared to vanilla generation in the limit, and a priori, it would appear that only a strict subset of countable collections may admit non-uniform generation. Nevertheless, as shown independently by both Li et al. (2024) and Charikar and Pabbaraju (2024), non-uniform generation in the limit also turns out to be possible for every countable collection!

Given these strong positive results, a natural direction that has not been investigated yet is to determine the *optimal* non-uniform generation algorithm, which requires the least possible number of inputs for *every language simultaneously* in the collection. However, for any language L, there is an algorithm that achieves $t(L) = 1$. Such an algorithm simply generates strings from L at the outset, until it sees a string outside L. The algorithms of Li et al. (2024) and Charikar and Pabbaraju (2024) (see Section 2.1) can do this for any language L. Since the optimal generation time for any language L is 1, achieving optimality simultaneously for all languages L is not feasible.

Nevertheless, in this work, we initiate a study of *Pareto-optimal* non-uniform language generation. We say that a sequence of generation times $t(L_1), t(L_2), \ldots$ for languages in the collection is Pareto-optimal, if it satisfies the following property: any algorithm whose generation time for some L_i is smaller than $t(L_i)$ *must* satisfy that its generation time for some other language L_j is larger than $t(L_j)$.[1] As our first contribution, we give a constructive process to obtain a canonical Pareto-optimal sequence of generation times for any given collection. The process resembles the textbook insertion sort algorithm, albeit with a carefully chosen comparator, and incrementally computes an ordering of the first n languages in the collection, for increasing n. Maintaining the ordering given by the comparator ensures that the generation time for every language is determined Pareto-optimally.

The previous non-uniform generation algorithms of Li et al. (2024) and Charikar and Pabbaraju (2024) do not achieve Pareto-optimality (see Example 1). Taking inspiration from our constructive process, we next give a non-uniform generation algorithm that is *almost* Pareto-optimal, in the following sense: the generation times of the algorithm can be made to match the generation times in the canonical Pareto-optimal sequence constructed above for an arbitrarily large (but finite) prefix of languages in the collection.

Theorem 1 (Almost Pareto-optimal Non-Uniform Language Generation) *Given a collection $\mathcal{C} = (L_1, L_2, \ldots)$, for any $n < \infty$, there exists an algorithm that non-uniformly generates from $\mathcal{C}$, and its sequence of generation times $t(L_1), t(L_2), \ldots$ for languages in $\mathcal{C}$ satisfies the following: any other algorithm whose generation time for some language L_i for $i \leq n$ is smaller than $t(L_i)$, must satisfy that its generation time for some other language L_j (also for $j \leq n$) is larger than $t(L_j)$.*

We also identify a technical condition on a collection that is sufficient to guarantee that the algorithm from the theorem above attains Pareto-optimality for the *entire* collection (Theorem 11);

1. This is a slight abuse of notation, since our definition of Pareto-optimal sequence of generation times does not require an algorithm which *achieves* this sequence. Technically, this is a lower bound on the Pareto front.

that is, its sequence of generation times *entirely* matches the Pareto-optimal sequence given by our constructive process, as opposed to matching it on just a finite prefix. Could the sufficient condition also be necessary? Is it possible to achieve Pareto-optimal generation times for all countable collections? Towards this, we identify two simple collections that do not satisfy our sufficient condition (Proposition 12 and Proposition 13) for which it is *impossible* for any algorithm to attain a Pareto-optimal sequence of generation times for the entire collection. We leave open the tantalizing question of obtaining an exact characterization of collections that admit Pareto-optimal non-uniform generation for future work.

Finally, we demonstrate that our framework above provides a *generic blueprint* for obtaining Pareto-optimal algorithms, even under other specialized settings requiring stronger guarantees than standard non-uniform generation. Namely, we study the recently introduced settings of *noisy* (Raman and Raman, 2025) and *representative* (Peale, Raman, and Reingold, 2025) generation. We show how suitably modifying the comparator in our insertion sorting process makes both these settings fit into our framework, allowing us to obtain almost Pareto-optimal non-uniform generation algorithms for both these settings. Lastly, we derive sufficient conditions as above for when our algorithms attain exact Pareto-optimality in these settings.

Significance of Results. Given the extremely complex nature of language models in practice, a sound theoretical model that helps understand the fundamental reasons for why language generation is tractable is valuable. In this regard, the model of generation in the limit by Kleinberg and Mullainathan (2024) has proved to be a useful sandbox for rigorously analyzing practical phenomena in language models. For example, in this model, it is possible to rigorously argue why hallucinations are necessary for generative models that are "expressive enough" (Kalavasis et al., 2025; Charikar and Pabbaraju, 2024; Kalavasis et al., 2024).

Our work significantly develops a notion of "optimality" in this model, which was previously missing. Even after the appropriate definition of Pareto-optimality has been established, important questions remain: 1) What even is a provably valid sequence of Pareto-optimal generation times that can be treated as an attainable benchmark? 2) How does one obtain an algorithm that achieves Pareto-optimality? Our work introduces new techniques to address both these questions. For (1), we construct a Pareto-optimal sequence through an insertion sort procedure (Procedure 1), which reorders languages in the collection according to a carefully chosen criterion: finite intersections of languages in prefixes of the collection. For (2), we need to appropriately blend in the algorithm of Charikar and Pabbaraju (2024) into the insertion sort process, in a way that it inherits the Pareto-optimal generation times established in (1). Both these steps need intricate arguments in order to achieve the desired guarantee of Pareto-optimality. It is worth noting that prior to our work, such Pareto-optimal generation was not known even for finite collections. Our impossibility results in Section 3.4 introduce an interesting notion of "admissible sequences" for generation: these are an assignment of generation times to languages that can be validly attained by a generation algorithm. It is an intriguing open direction to characterize admissible sequences in full generality, and also general conditions when Pareto-optimality is impossible.

To summarize, we hope that insights from our work, and in general from a rigorous understanding of the model of generation in the limit, can shed light on empirical mysteries, and also inspire practical algorithms, which can, for example, make (Pareto) optimal use of resources (samples, compute, etc.).

2. Preliminaries

We first describe the setup of language generation in the limit introduced by Kleinberg and Mullainathan (2024). In this model, there is an underlying countable collection of languages, where each language is a countably infinite subset of a countably infinite universe U. We use the notation $(L_1, L_2, \ldots)$ to denote the languages in a countable collection $\mathcal{C}$; we use this notation, in lieu of curly brackets $\{\}$, to emphasize that the ordering is important, and that there might be duplicates of a language at multiple indices.

An *enumeration* of a language L comprises of a sequence $x_1, x_2, \ldots$ of strings (repetitions allowed) such that $x_t \in L$ for every $t \geq 1$, and furthermore, for every $x \in L$, there exists a finite t such that $x_t = x$. The generation in the limit game operates as follows: an adversary chooses a language from the known collection $\mathcal{C}$, and proceeds to enumerate it in a worst-case fashion. At each time step t, after having observed the sequence $x_1, \ldots, x_t$ enumerated so far by the adversary, the task of a generating algorithm is to generate a string z_t. Note that the input does not comprise of any strings outside the chosen language, and the algorithm gets no feedback about the string it generates at any time step. We denote the set of *distinct* strings in the input sequence $x_1, \ldots, x_t$ enumerated up until time t by S_t. The success criterion for the algorithm is defined as follows:

Definition 2 (Generation in the Limit (Kleinberg and Mullainathan, 2024)) *A generating algorithm algorithm $\mathcal{G}$ generates in the limit from a collection $\mathcal{C} = (L_1, L_2, \ldots)$ if for every $L \in \mathcal{C}$ and every enumeration σ of L, there exists a finite $t(L, \sigma)$[2] such that for every $t \geq t(L, \sigma)$, the string z_t generated by the algorithm at time step t belongs to $L \setminus S_t$.*

The collection above plays the role of the "function class" from which a generating algorithm is trying to learn. For example, we can imagine that each language in the collection represents the set of strings that a language model may generate corresponding to a given configuration of its parameters, and as more inputs get revealed with time, the model tunes its parameters, simultaneously moving around in the collection. To make this connection slightly more precise, note that modern language models are trained by performing some form of gradient-based optimization on the weights of a transformer model of a fixed architecture, after seeing a finite batch of samples. In order to model this, consider the language collection consisting of one language for every possible assignment of the transformer weights. Indeed, with finite precision for the weights of the transformer, observe that the set of all possible transformer models can be enumerated in a countable collection. The gradient-based training process can then be seen as traversing this collection in a way that optimizes the target objective. In this sense, the setup of generating from a countable language collection does in principle capture the practice of learning a language model.

The general positive result of Kleinberg and Mullainathan (2024) shows that generation in the limit can achieved for *every* countable collection of languages. Observe however that the time for valid generation in the definition above, namely $t(L, \sigma)$, is a function of both the target language L as well as the enumeration order σ. The notion of non-uniform language generation tries to get rid of the dependence on the enumeration order σ in the success time $t(L, \sigma)$.

Definition 3 (Non-uniform Generation (Li et al., 2024)) *A generating algorithm algorithm $\mathcal{G}$ non-uniformly generates in the limit from a collection $\mathcal{C} = (L_1, L_2, \ldots)$ if for every $L \in \mathcal{C}$, there exists a*

2. We suppress the implicit dependence on the underlying collection $\mathcal{C}$ for convenience.

$t(L)$ such that for every enumeration σ of L presented to $\mathcal{G}$, the string z_t generated by the algorithm at time step t belongs to $L \setminus S_t$ for all t satisfying $|S_t| \geq t(L)$.

As it turns out, the stronger guarantee of non-uniform generation in the limit can also be achieved for every countable collection of languages, as shown by Li et al. (2024) and Charikar and Pabbaraju (2024). It helps to think about the language-dependent success time $t(L)$ as measuring how easy it is to discern L from other languages in the collection: the larger $t(L)$ is, the larger the amount of resources—number of inputs, computation—that an algorithm must invest on language L in order to generate validly from it. We can then ask: what is the minimal amount of resources that can be assigned to every language in the collection? Towards this, we propose the following definition of Pareto-optimality:

Definition 4 (Pareto-optimality) *A sequence $t(L_1), t(L_2), \ldots$ for languages in $\mathcal{C}$ is Pareto-optimal if any algorithm $\mathcal{G}$ that non-uniformly generates from $\mathcal{C}$ and satisfies that $t_{\mathcal{G}}(L_i) < t(L_i)$ for some i, must satisfy that its generation time for some other L_j satisfies $t_{\mathcal{G}}(L_j) > t(L_j)$*

In our study of this guarantee, the notion of *finite intersections* between languages will play a key role. A subcollection $\mathcal{C}'$ of $\mathcal{C}$ has finite intersection if $|\cap_{L\in\mathcal{C}'} L| < \infty$. The subcollection $\mathcal{C}'$ of $\mathcal{C}$ that has *largest* finite intersection satisfies

$$\mathcal{C}' \in \arg\max_{\mathcal{C}''}\{|\cap_{L\in\mathcal{C}''} L| : \mathcal{C}'' \subseteq \mathcal{C}, |\cap_{L\in\mathcal{C}''} L| < \infty\}.$$

2.1. Prior Guarantees

We now formally review the non-uniform generation guarantees of the algorithms given by Li et al. (2024) and Charikar and Pabbaraju (2024). We will specify the generation time required by these algorithms for each language L_i in the collection $\mathcal{C} = (L_1, L_2, \ldots)$.

Guarantee of Li et al. (2024). Define the *closure dimension* of a collection $\mathcal{C}$ to be the size of the largest finite intersection of a subcollection of $\mathcal{C}$:[3]

$$d := \max_{\mathcal{C}'}\{|\cap_{L\in\mathcal{C}'} L| : \mathcal{C}' \subseteq \mathcal{C}, |\cap_{L\in\mathcal{C}'} L| < \infty\}^{4}, \tag{1}$$

For every language L_i, consider the closure dimension d_i of the prefix collection $(L_1, \ldots, L_i)$. Observe that the sequence $\{d_i\}_{i\geq 1}$ is non-decreasing. The guarantee for the algorithm given by Li et al. (2024) is as follows: If $\lim_{i\to\infty} d_i = \infty$, then the generation time for each language is $t(L_i) = d_i + 1$. If $\lim_{i\to\infty} d_i = c < \infty$, then $t(L_i) = \max(i, c+1)$.

Guarantee of Charikar and Pabbaraju (2024). For any L_i, consider the largest finite intersection of a subcollection of $(L_1, \ldots, L_i)$ *that contains* L_i, called the *non-uniform complexity* of L_i:

$$m(L_i) := \max_{\mathcal{C}'}\{|\cap_{L\in\mathcal{C}'} L| : \mathcal{C}' \subseteq (L_1, \ldots, L_i), \mathcal{C}' \ni L_i, |\cap_{L\in\mathcal{C}'} L| < \infty\}. \tag{2}$$

3. Li et al. (2024) originally define the closure dimension as the size of the largest set T such that $|\cap_{L\in\mathcal{C}, T\subseteq L} L| < \infty$; the definition stated here is equivalent.
4. For concreteness, we define the max over an empty set to be 0, and the arg max to be null.

Notice that the difference in d_i and $m(L_i)$ above is that the latter quantity is computed as the maximum over a smaller set (due to the additional condition that $\mathcal{C}' \ni L_i$); hence, $m(L_i) \leq d_i$. The guarantee given by Theorem 6 in Charikar and Pabbaraju (2024) is that $t(L_i) = \max(i, m(L_i)+1)$.

We also observe that the generation time guarantee of Charikar and Pabbaraju (2024) for the language in the *first* position in $\mathcal{C}$ is *always* 1, which is the smallest achievable generation time. So, for any other algorithm, if the generation time for some language L in the collection is larger than 1, we can place L at the start of $\mathcal{C}$, and run the algorithm of Charikar and Pabbaraju (2024) to get an improved generation time of 1 for L. Thus, no algorithm can simultaneously achieve the smallest possible generation time for *every* language L, unless its generation time for every language is 1.

2.2. Other Related Work

The work of Kleinberg and Mullainathan (2024) also initiated the study of language generation in the limit with a *uniform* guarantee, albeit for finite collections. In uniform generation, the generation time of an algorithm must be a global quantity for the collection, irrespective of the language being enumerated by the adversary. Recall that in non-uniform generation, the generation time may depend on the language being enumerated (but in neither setting may it depend on the enumeration order). Li et al. (2024) formalized uniform generation more generally for collections that are not necessarily finite, and obtained a quantitatively tight characterization of the uniform generation time for a collection in terms of its closure dimension. Both these works also study language generation in the limit with a prompt. More recently, Hanneke et al. (2025) and Bai et al. (2025) established results on the behavior of non-uniform generation under unions of language collections. Bai et al. (2025) also study several other variants of generation, including generation with feedback, which was introduced by Charikar and Pabbaraju (2024).

Besides uniform/non-uniform generation, there has also been interest in understanding the *breadth* of the target language covered by algorithms that generate in the limit. In particular, Charikar and Pabbaraju (2024), Kalavasis et al. (2025) and Kalavasis et al. (2024) relate definitions of generating with breadth to language identification, and establish strong negative results for the same. Very recently, Kleinberg and Wei (2025) proposed more fine-grained *density* measures for breadth. Their work proposes new algorithms for generation with a view to maximize these density measures, and has remarkable positive results.

3. Pareto-optimal Non-uniform Generation

We begin with a motivating example of a simple collection $\mathcal{C}$ for which the guarantees for both the algorithms of Li et al. (2024) and Charikar and Pabbaraju (2024) get Pareto-dominated by the latter algorithm when run on a suitable *reordering* of $\mathcal{C}$.

Example 1 *Consider the collection $\mathcal{C} = (L_1, L_2, L_3, \ldots)$, where*

$$L_1 = \{1, \ldots, 99, 100\} \cup \{-p_1, -p_1^2, -p_1^2, -p_1^3, \ldots\}$$
$$L_2 = \{1, \ldots, 99, 100\} \cup \{101, \ldots, 199, 200\} \cup \{201, \ldots, 299, 300\} \cup \{-p_2, -p_2^2, -p_2^2, -p_2^3, \ldots\}$$
$$L_3 = \{101, \ldots, 199, 200\} \cup \{-p_3, -p_3^2, -p_3^2, -p_3^3, \ldots\}$$
$$L_4 = \{201, \ldots, 299, 300\} \cup \{-p_4, -p_4^2, -p_4^2, -p_4^3, \ldots\}$$
$$L_i = \{-p_i, -p_i^2, -p_i^2, -p_i^3, \ldots\}, \quad \forall i \geq 5.$$

Here, p_n refers to the n^{th} prime number.

Denoting the closure dimension of the prefix $(L_1, \ldots, L_i)$ by d_i as in Section 2.1, we observe that $d_1 = 0$, and $d_i = 100$ for $i \geq 2$ (and hence $\lim_{i\to\infty} d_i = 100$). The guarantee of Li et al. (2024) then gives

$$t(L_i) = 101 \quad \text{for } 1 \leq i \leq 100, \qquad \text{and} \qquad t(L_i) = i, \ \forall i \geq 101.$$

Similarly, we observe that $m(L_1) = 0$, $m(L_2) = 100$, $m(L_3) = 100$, $m(L_4) = 100$, and $m(L_i) = 0$ for $i \geq 5$. The guarantee of Charikar and Pabbaraju (2024) then gives

$$t(L_1) = 1, t(L_2) = 101, t(L_3) = 101, t(L_4) = 101, \qquad \text{and} \qquad t(L_i) = i, \ \forall i \geq 5.$$

But now, consider reordering the languages in $\mathcal{C}$, so that L_2 is *after* L_3 and L_4. That is, set $\mathcal{C} = (L_1, L_3, L_4, L_2, L_5, L_6, \ldots)$, and let us now consider the guarantee of Charikar and Pabbaraju (2024) for this reordered collection. We obtain that $m(L_1) = 0$, $m(L_3) = 0$, $m(L_4) = 0$, $m(L_2) = 100$, and $m(L_i) = 0$ for $i \geq 5$. So, we obtain the following generation times:

$$t(L_1) = 1, t(L_2) = 101, t(L_3) = 1, t(L_4) = 1, \qquad \text{and} \qquad t(L_i) = i, \ \forall i \geq 5.$$

Observe that this sequence of generation times is *strictly better* than both the above guarantees; in particular, the generation time for L_3 and L_4 is strictly smaller, while the generation time for any other language is no larger. The example above can be easily modified to an instance where $\lim_{i\to\infty} d_i = \infty$, or to make the suboptimality appear more extreme.

The structure in the example above motivates the following procedure for constructing a Pareto-optimal sequence of generation times.

3.1. Pareto-optimal Sequence of Generation Times

We specify Procedure 1 which computes a non-uniform complexity $m^\star(L_i)$ for every $i \geq 1$. In the i^{th} iteration of this procedure, we determine $m^\star(L_i)$, and also a subcollection $\mathcal{C}(L_i)$, which is a "witness" for $m^\star(L_i)$. In the process, we maintain an ordering $\mathcal{C}'_i$ of the first i languages in $\mathcal{C}$ (which may be different from the way these languages are ordered in $\mathcal{C}$). This ordering $\mathcal{C}'_i$ is maintained using insertion sort: in the i^{th} iteration, we first append the i^{th} language L_i in $\mathcal{C}$ to the ordering $\mathcal{C}'_{i-1}$ of the first $i-1$ languages that was determined in the previous iterations. Then, we advance the position of L_i in the ordering one position at a time until a certain local condition is satisfied.

Some remarks about the procedure are in order. We note that L'_j is always equal to L_i at the beginning of every iteration of the while loop. For every i, $\mathcal{C}(L_i)$ and $m^\star(L_i)$ are assigned once and for all at the end of the i^{th} iteration of the for loop; $\mathcal{C}(L_i)$ is set to some subcollection of $(L_1, \ldots, L_i)$ that contains L_i and has finite intersection (if it is non-empty), and $m^\star(L_i) = |\cap_{L\in\mathcal{C}(L_i)} L|$. Furthermore, at the end of the i^{th} iteration of the procedure, insertion sort guarantees that the sequence $m^\star(L'_1), m^\star(L'_2), \ldots, m^\star(L'_i)$ is non-decreasing. However, the sequence $m^\star(L_1), m^\star(L_2), m^\star(L_3), \ldots$ is *not* necessarily non-decreasing. We also observe the following:

Observation 5 (Languages in witness have smaller complexity) *For every $i \geq 1$, when $\mathcal{C}(L_i)$ and $m^\star(L_i)$ are determined in Step(c) above, $\mathcal{C}(L_i)$ is either empty, or contains at least one $L_j \neq L_i$ for $j < i$, and furthermore, every $L_j \neq L_i$ in $\mathcal{C}(L_i)$ satisfies that $m^\star(L_j) < m^\star(L_i)$.*

Procedure for Computing Non-uniform Complexities $m^\star(L_i)$

1. Initialize $\mathcal{C}'_0 = ()$.
2. For $i = 1, 2, 3, \ldots$
   (a) Initialize $\mathcal{C}'_i$ by appending L_i at the end of $\mathcal{C}'_{i-1}$, and denote the resulting sequence $\mathcal{C}'_i$ as $(L'_1, \ldots, L'_i)$. Note that this is simply a permutation of $(L_1, \ldots, L_i)$. We want to run one iteration of "insertion sort" so that L_i is at its correct position in $\mathcal{C}'_i$. Initialize $j = i$.
   (b) While true do:
      i. Let $\mathcal{C}_{\text{check}}$ be the subcollection of $(L'_1, \ldots, L'_j)$ that has the largest finite intersection among subcollections that contain L'_j. If there is no such subcollection, set $\mathcal{C}_{\text{check}} = ()$, $m_{\text{check}} = 0$. Else, set $m_{\text{check}} = |\cap_{L \in \mathcal{C}_{\text{check}}} L|$.
      ii. If $j \leq 1$ or $m_{\text{check}} > m^\star(L'_{j-1})$, break.
      iii. Swap L'_j and L'_{j-1} in $\mathcal{C}'_i$.
      iv. $j \leftarrow j - 1$
   (c) Set $\mathcal{C}(L_i) = \mathcal{C}_{\text{check}}$, $m^\star(L_i) = m_{\text{check}}$.

Procedure 1: Insertion Sort for Non-uniform Generation

Proof Suppose $\mathcal{C}(L_i)$ is not empty when it is determined. Note that this must mean that $j > 1$ when the while loop was exited. Note also that $\mathcal{C}(L_i)$ can't only contain copies of L_i, since in this case, the intersection of languages in $\mathcal{C}(L_i)$ (simply L_i) would be infinite (by assumption that all languages are infinite). Observe then that by virtue of the while loop exiting, it must have been the case that $m_{\text{check}} > m^\star(L'_{j-1})$. But recall that $m^\star(L'_1), \ldots, m^\star(L'_{j-1})$ is a non-decreasing sequence. The claim then follows since any language L_j in $\mathcal{C}(L_i)$ (which, recall, is assigned to be $\mathcal{C}_{\text{check}}$) that is not equal to L_i is contained in $(L'_1, \ldots, L'_{j-1})$, and the fact that we assign $m^\star(L_i) = m_{\text{check}}$. ■

This observation ensures that $m^\star(\cdot)$ forms a Pareto-optimal sequence of generation times.

Claim 6 ($m^\star(\cdot)$ forms a Pareto-optimal sequence) *Any algorithm $\mathcal{G}$ that satisfies $t_{\mathcal{G}}(L_i) < m^\star(L_i) + 1$ for some L_i must have $t_{\mathcal{G}}(L_j) > m^\star(L_j) + 1$ for some $j < i$.*

Proof Note that the language L_i must be such that $m^\star(L_i) > 0$, since $t_{\mathcal{G}}(\cdot)$ is always at least 1.But this means that $\mathcal{C}(L_i) \neq ()$. Then, from Observation 5, it must be the case that at least some $L_j \in \mathcal{C}(L_i)$ is not equal to L_i, where $j < i$. Suppose an adversary enumerates all the $m^\star(L_i)$ strings in $I = \cap_{L \in \mathcal{C}(L_i)} L$ in the first $m^\star(L_i)$ time steps. Let z be the string generated by the algorithm $\mathcal{G}$ at $t = m^\star(L_i)$. By the guarantee of $\mathcal{G}$ that $t_{\mathcal{G}}(L_i) < m^\star(L_i) + 1$, it must be the case that $z \in L_i \setminus I$ (otherwise, the adversary can continue to enumerate the rest of L_i beyond this point, and we would have obtained an enumeration of L_i for which $\mathcal{G}$ violates its guarantee for L_i at $t = m^\star(L_i)$). But since we have enumerated every string in the intersection of languages in $\mathcal{C}(L_i)$, it must be the case that $z \notin L_j$ for some $L_j \in \mathcal{C}(L_i)$ where $L_j \neq L_i$ and $j < i$. But note also that from Observation 5, $m^\star(L_j) < m^\star(L_i)$. That is, if the adversary were to continue enumerating L_j beyond t, and $t_{\mathcal{G}}(L_j)$

were at most $m^\star(L_j)+1$, which is at most $m^\star(L_i)$, then the string z generated by $\mathcal{G}$ at this time should have belonged to L_j, which we know is not the case. Thus, $t_{\mathcal{G}}(L_j) > m^\star(L_j)+1$. ■

We conclude this section by discussing the computational cost of Procedure 1. Similar to Charikar and Pabbaraju (2024), the procedure requires access to two kinds of oracles: (1) an oracle that, given a finite collection of languages, responds with whether the intersection of languages in the collection is finite, and (2) an oracle, which answers membership queries—namely, queries of the form "is x in L_j?" for any choice of x and language L_j in the collection. At step i of the insertion sort when language L_i is processed, the procedure requires roughly 2^i queries to oracle (1) to determine the largest finite intersection, and place language L_i at its correct position. The rest of the computation is the same as that in the original algorithm of Charikar and Pabbaraju (2024), namely, a forward pass to construct an infinite intersection of languages consistent with the input, which requires polynomially many queries to oracles (1) and (2).

3.2. (Almost) Pareto-optimal Non-uniform Generation Algorithm

While Procedure 1 specifies a sequence of Pareto-optimal generation times, it also motivates a corresponding algorithm that *almost* attains this sequence. To begin, let us think about what happens when we swap L'_j and L'_{j-1} within an iteration of insertion sort. Note that after moving L'_j to position $j-1$, the collection $\mathcal{C}_{\text{check}}$ is obtained as the max over subcollections that are *strictly* contained among the subcollections under consideration before moving; hence, it is clear that m_{check} can only decrease. More importantly, we also have the following nice property, which turns out to be crucial for the correctness of the algorithm we state below: if one were to compute m_{check} for L'_{j-1} after it has been moved forward to position j, it remains equal to $m^\star(L'_{j-1})$, *even if* m_{check} is now a maximum over a strictly larger space—this is a consequence of the while loop not exiting.

Claim 7 (Arg max Maintained) *For every $i \geq 1$, and for every $t \geq i$, suppose k is the index of language L_i in $\mathcal{C}'_t = (L'_1, \ldots, L'_t)$ at the end of the t^{th} iteration of the for loop. Then,*

$$\mathcal{C}(L_i) \in \arg\max_{\mathcal{C}_{\text{check}}} \{|\cap_{L\in\mathcal{C}_{\text{check}}} L| : \mathcal{C}_{\text{check}} \subseteq (L'_1, \ldots, L'_k), \mathcal{C}_{\text{check}} \ni L'_k, |\cap_{L\in\mathcal{C}_{\text{check}}} L| < \infty\}. \quad (3)$$

Proof For $t = i$, the claim is true by definition of $\mathcal{C}(L_i)$. Now suppose as our inductive hypothesis that the claim is true for some $t \geq i$. We will show that it is true for $t+1$. Suppose k is the index of L_i at the end of the t^{th} iteration of the for loop. If k continues to be the index of L_i at the end of the $(t+1)^{\text{th}}$ iteration as well (meaning that the language L_{t+1} was placed at some index larger than k in $\mathcal{C}'_{t+1}$), the claim continues to be true simply by the inductive hypothesis. Otherwise, the language L_{t+1} was placed at an index in $\{1, \ldots, k\}$, and so L_i was moved to index $k+1$ in $\mathcal{C}'_{t+1}$. In this case, consider the precise iteration of the while loop when j was equal to $k+1$ (so that L'_{k+1} was L_{t+1}), and m_{check} was found to be at most $m^\star(L'_k)$ in Step ii. At this point, Step iii tells us to swap L'_k and L'_{k+1} (this will make L'_{k+1} equal to L_i, and L'_k equal to L_{t+1}). Thereafter, note that the (unordered) subcollection of languages before L'_{k+1} remains unchanged until the while loop breaks, and the present iteration of the for loop completes. So, it suffices to argue that, *before* swapping,

$$\mathcal{C}(L_i) \in \arg\max_{\mathcal{C}_{\text{check}}} \{|\cap_{L\in\mathcal{C}_{\text{check}}} L| : \mathcal{C}_{\text{check}} \subseteq (L'_1, \ldots, L'_k, L'_{k+1}), \mathcal{C}_{\text{check}} \ni L'_k, |\cap_{L\in\mathcal{C}_{\text{check}}} L| < \infty\}.$$

Recall that our inductive hypothesis already guarantees that $\mathcal{C}(L_i)$ belongs to the arg max over all feasible $\mathcal{C}_{\text{check}}$, where the condition on $\mathcal{C}_{\text{check}}$ only considers subcollections of $(L'_1, \ldots, L'_k)$. That

is, the only additional language now being considered in the condition is L'_{k+1}. So, if the max were to increase from what it was when L'_{k+1} was not being considered, it must be the case that the $\mathcal{C}_{\text{check}}$ which is the new $\arg\max$ *contains* L'_{k+1}. But this means that before swapping, there exists a subcollection of $(L'_1, \ldots, L'_k, L'_{k+1})$ that contains L'_{k+1}, and has finite intersection of size strictly larger than $|\cap_{L \in \mathcal{C}(L_i)} L|$. This contradicts the fact that m_{check} was found to be at most $m^\star(L'_k)$ (since $L'_k = L_i$, and $|\cap_{L \in \mathcal{C}(L_i)} L| = m^\star(L_i)$). Thus, it must be the case that $\mathcal{C}(L_i)$ continues to be in the $\arg\max$ even after swapping. This completes the proof of the inductive step. ■

Claim 7 motivates the following algorithm for Pareto-optimal non-uniform generation:

Algorithm. Fix some non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ that satisfies $\lim_{t\to\infty} f(t) = \infty$. At time step t, the algorithm follows $f(t)$ iterations of Procedure 1 for the purpose of constructing the ordered collection $\mathcal{C}'_{f(t)} = (L'_1, \ldots, L'_{f(t)})$ (a permutation of the first $f(t)$ languages in $\mathcal{C}$). It then initializes $I_t = ()$, and traverses $L'_1, L'_2, \ldots, L'_{f(t)}$ in this order. Whenever it encounters a language L'_j that satisfies $S_t \subseteq L'_j$, it checks if $|\cap_{L \in I_t \cup \{L'_j\}} L| = \infty$. If so, it appends L'_j to I_t; otherwise, it moves on, leaving I_t unaffected. Observe that at the end of the traversal, I_t is maintained to be a collection of languages that has infinite intersection (if it is non-empty). So, at the end of its traversal, if I_t is empty, the algorithm generates an arbitrary new string from the universe U, and if I_t is non-empty, the algorithm generates a new string from $\cap_{L \in I_t} L$.

We note that the algorithm of Charikar and Pabbaraju (2024) is essentially the algorithm above, with the choice of $f(t) = t$, and $\mathcal{C}'_{f(t)} = (L_1, \ldots, L_{f(t)})$ (i.e., the default ordering in $\mathcal{C}$). The crucial change is the carefully maintained ordering $(L'_1, \ldots, L'_{f(t)})$. We can use Claim 7 to derive the following characterization of non-uniform generation times for the algorithm:

Theorem 8 (Non-uniform Generation Times) *The algorithm described above non-uniformly generates from the collection $\mathcal{C} = (L_1, L_2, \ldots)$, with $t^\star(L_i) = \max(g(i), m^\star(L_i) + 1)$, where $g(i)$ is the smallest number j for which $f(j) \geq i$.*

Proof Suppose that the target language being enumerated by the adversary is L_i. Consider any time t for which $|S_t| \geq t^\star(L_i)$; this implies $t \geq t^\star(L_i) \geq g(i)$. Since f is non-decreasing, the language L_i is under consideration by the algorithm at time t. So, let k be the index of the language L_i in the collection $\mathcal{C}'_{f(t)} = (L'_1, \ldots, L'_{f(t)})$ constructed by the algorithm at time step t. We only need to argue that when L'_k is encountered by the algorithm as it traverses $\mathcal{C}'_{f(t)}$, it finds that $|\cap_{L \in I_t \cup \{L'_k\}} L| = \infty$ (note that $S_t \subseteq L'_k$ holds, since L'_k is the target language being enumerated). Assume for the sake of contradiction that $|\cap_{L \in I_t \cup \{L'_k\}} L| < \infty$. In this case, observe that $I_t \cup \{L'_k\}$ is a subcollection of $(L'_1, \ldots, L'_k)$ which contains L'_k and has finite intersection. Furthermore, the size of this intersection is at least $m^\star(L_i) + 1$ since every language in the subcollection contains S_t by definition of the algorithm. This would imply that the maximum in the RHS of (3) in Claim 7 is at least $m^\star(L_i) + 1$, which contradicts $\mathcal{C}(L_i)$ realizing the maximum, since $|\cap_{L \in \mathcal{C}(L_i)} L| = m^\star(L_i)$. Thus, $L'_k = L_i$ gets added to I_t when it is encountered by the algorithm; since the algorithm generates a new string from $\cap_{L \in I_t} L$ which is ensured to be an infinite set, we are done. ■

Remark 9 *The guarantee above is strictly no worse than the non-uniform generation guarantee of Charikar and Pabbaraju (2024) (see Section 2.1), if we substitute $f(t) = t$.*

Remark 10 *Observe that by choosing f to be arbitrarily fast-growing, we can ensure that $t^\star(L_i) = m^\star(L_i) + 1$ for all $i \leq n$, for an arbitrarily large n. In this sense, and as also stated in Theorem 1, our algorithm above almost attains Pareto-optimality, since it can match the generation times of a Pareto-optimal sequence (Claim 6) in an arbitrarily large prefix .*

3.3. Sufficient Condition for Exact Pareto-optimality

We identify a sufficient condition for a collection that guarantees *exact* Pareto-optimality for our algorithm. Here, for an appropriately chosen function f, our algorithm matches the Pareto-optimal sequence of generation times identified by the procedure above (Claim 6) at *every* language.

Theorem 11 (Sufficient Condition for Exact Pareto-optimality) *Let $\mathcal{C} = (L_1, L_2, \ldots)$ be a collection that satisfies the following property: for every $t \in \mathbb{N}$, $|\{j : m^\star(L_j) + 1 \leq t\}| < \infty$, where the $m^\star(\cdot)$ values are those computed in Procedure 1. Then, there exists an algorithm that achieves a Pareto-optimal sequence of non-uniform generation times for $\mathcal{C}$.*

Proof Consider $f : \mathbb{N} \to \mathbb{N}$ defined as $f(t) = \max\{j : m^\star(L_j) + 1 \leq t\}$. Observe that f is a non-decreasing function, and that $\lim_{t\to\infty} f(t) = \infty$. Furthermore, observe also that for any i, $f(m^\star(L_i) + 1) \geq i$. Therefore, if $g(i)$ is the smallest number j for which $f(j) \geq i$, then $g(i) \leq m^\star(L_i) + 1$. So, if we choose to run the algorithm given in Section 3.2 with this function f, the guarantee of Theorem 8 implies $t^\star(L_i) = m^\star(L_i) + 1$ for every L_i. From Claim 6, we conclude that the algorithm is Pareto-optimal. ■

A direct corollary of Remark 10 and Theorem 11 is that exact Pareto-optimal non-uniform generation can be achieved for every *finite* collection of languages.

3.4. Impossibility of Pareto-optimality

It is natural to wonder if the sufficient condition stated above is also necessary: this would exactly characterize the collections for which achieving Pareto-optimality is possible; moreover, our algorithm above would achieve such Pareto-optimality. Towards this, we exhibit two collections, for which we show that there *cannot* exist an algorithm that achieves a Pareto-optimal sequence of generation times. One of these collections can be non-uniformly, but not uniformly generated, while the other can even be *uniformly* generated. Both these collections *fail* the sufficient condition stated in Theorem 11. We leave open the complete characterization of Pareto-optimality for future work.

Proposition 12 (Non-uniformly Generatable, No Pareto-optimality) *No algorithm can achieve Pareto-optimal non-uniform generation times for the collection $\mathcal{C} = \{\mathbb{Z} \setminus \{i\}\}_{i\in\mathbb{Z}}$.*

Proof We note first that the collection in question cannot be uniformly generated because its closure dimension (see Section 2.1) is unbounded. To see this, observe that for any finite $S \subseteq \mathbb{Z}$, the intersection of languages in $\mathcal{C}$ that contain S is precisely S. However, this collection is non-uniformly generatable, since it is countable.

Let $L_{e_1}, L_{e_2}, L_{e_3}, \ldots$ be an enumeration of all the languages in $\mathcal{C}$ (with no repetition), where $L_{e_i} = \mathbb{Z} \setminus \{e_i\}$ for $e_i \in \mathbb{Z}$. Let $t_1, t_2, t_3, \ldots$ be a sequence of candidate generation times for these languages, where every t_i is finite and at least 1. We call the sequence *admissible* if it satisfies the following property: for every $t \geq 1$, the size of the set $\{i : t_i \geq t\}$ is infinite. We claim that there

exists an algorithm that achieves the generation times given by the sequence $t_1, t_2, \ldots$ if and only if it is admissible.

For one direction, suppose that the sequence is not admissible, meaning that there exists some t for which $|\{i : t_i \geq t\}| < \infty$. Suppose that an algorithm achieves the generation times given in the sequence. Then, observe that the said algorithm *uniformly* generates from the collection, with a uniform time bound of $\max\{t_i : t_i \geq t\}$. Since we argued above that the collection cannot be uniformly generated, such an algorithm cannot exist.

For the other direction, suppose that the sequence is admissible. Consider the algorithm that operates as follows: at time step t, it obtains the subcollection $\mathcal{C}_t = \{L_{e_i} : t_i \leq t\}$. If $\mathcal{C}_t$ is non-empty, and $\cap_{L \in \mathcal{C}_t} L$ is also non-empty, the algorithm generates a new string from $\cap_{L \in \mathcal{C}_t} L$ (or repeats a string if there are no new strings available in this set); otherwise, it generates an arbitrary string. Note that this algorithm is oblivious to its input.

We claim that for every language $L_{e_i} \in \mathcal{C}$, this algorithm achieves the desired generation time of t_i. To see this, it suffices to argue that for every t that satisfies $|S_t| \geq t_i$, $\mathcal{C}_t$ contains L_{e_i}, and that $\cap_{L \in \mathcal{C}_t} L$ is infinite. Note first that any t satisfying $|S_t| \geq t_i$ immediately satisfies $t \geq t_i$ (since in t times steps, the input can have at most t distinct strings). By definition of the algorithm then, $\mathcal{C}_t$ contains L_{e_i}; in particular, it is non-empty.

Now, since the sequence is admissible, the set $\{j : t_j \geq t + 1\}$ is infinite. In particular, there exist infinitely many languages L_{e_j}, such that each L_{e_j} is excluded from $\mathcal{C}_t$. But observe that for any L_{e_j} that is excluded from $\mathcal{C}_t$, *every* language in $\mathcal{C}_t$ contains e_j. Since there are infinitely many such L_{e_j}, we conclude that $\cap_{L \in \mathcal{C}_t} L$ is infinite.

To finish the proof, assume for the sake of contradiction that there exists an algorithm that achieves non-uniform generation times $t_1, t_2, \ldots$ for $L_{e_1}, L_{e_2}, \ldots$ which are Pareto-optimal. By our reasoning above, the sequence $t_1, t_2, \ldots$ must be admissible. But now, consider any $t_i > 1$ in the sequence (such a t_i must exist by admissibility), and consider updating $t_i = 1$ (other generation times in the sequence are left unaffected). This updated sequence strictly Pareto-dominates the previous sequence. Furthermore, the updated sequence remains admissible, since we only changed one number. This means that there exists an algorithm that achieves the generation times in the updated sequence. Consequently, the algorithm we started with could not have been Pareto-optimal. ■

Proposition 13 (Uniformly Generatable, No Pareto-optimality) *No algorithm can achieve Pareto-optimal non-uniform generation times for the collection $\mathcal{C}$ comprising of languages $\{1, 2, \ldots, 100\} \cup \mathbb{Z}_{\geq i}$ for all $i \in \mathbb{Z}$, where $\mathbb{Z}_{\geq i} = \{i, i+1, i+2, \ldots\}$.*

We defer the proof of this result to Section A. We conclude this section by arguing that both the collections from Proposition 12 and Proposition 13 do not satisfy the sufficient condition of Pareto-optimality in Theorem 11. To see this, for either of these collections $\mathcal{C}$, note that the intersection of languages in any finite subcollection is infinite; since the $m^\star(\cdot)$ values in the Pareto-optimal sequence computed in Section 3.1 correspond to finite intersections of finite subcollections, we have that $m^\star(L_j) = 0$ for every $L_j \in \mathcal{C}$ (in fact, this holds true irrespective of the original ordering of $\mathcal{C}$ that we start with to compute the $m^\star(\cdot)$ values). Thus, the sufficient condition does not hold for $t = 1$, since $|\{j : m^\star(L_j) + 1 \leq 1\}| = \infty$.

We now proceed to study Pareto-optimality for two recent stylized models of generation: noisy and representative generation.

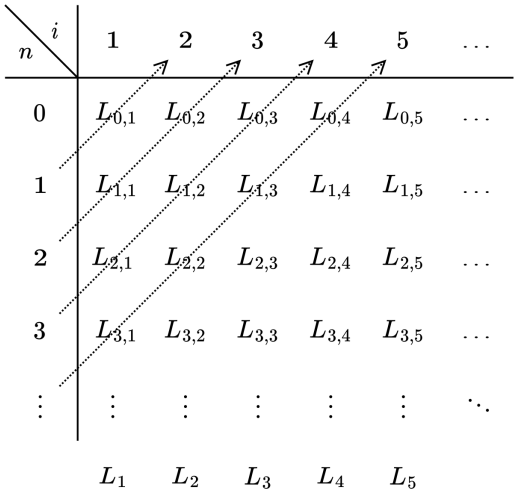

Figure 1: Diagonal traversal over languages arranged in a grid. The i^{th} column entirely consists of L_i. When we arrive at a copy of L_i in the n^{th} row (i.e., at $L_{n,i}$), we compute $m^\star_n(L_i)$.

4. Pareto-optimal Noisy Non-uniform Generation

In noisy generation (Raman and Raman, 2025), we consider *noisy enumerations* of the target language. For a language L, an enumeration of L at (finite) noise level $n \geq 0$ is a sequence $x_1, x_2, x_3, \ldots$, which satisfies: (1) For every $x \in L$, there exists $t < \infty$ for which $x_t = x$, and (2) $\sum_{t=1}^{\infty} \mathbb{1}[x_t \notin L] \leq n$.

Note that the noiseless setting considered in the previous section is only concerned with enumerations at noise level $n = 0$. For a set T, language L, and integer $a \geq 0$, we will say that L **a-contains T (or T is a-contained in L)** if $\sum_{x \in T} \mathbb{1}[x \notin L] \leq a$.

Definition 14 (Noisy Non-uniform Generation (Raman and Raman, 2025)) *An algorithm $\mathcal{G}$ noisily non-uniformly generates in the limit from a collection $\mathcal{C}$, if for every $L \in \mathcal{C}$ and every finite noise level $n \geq 0$, there exists $t_n(L)$ such that for every enumeration of L at noise level n presented to $\mathcal{G}$, the string z_t generated by $\mathcal{G}$ at time step t belongs to $L \setminus S_t$ for all t satisfying $|S_t| \geq t_n(L)$.*

Raman and Raman (2025) derive comprehensive results characterizing noisy (non-uniform as well as uniform) generation; however, examples like Example 1 render their algorithm to be Pareto-suboptimal as well; we will now work towards a Pareto-optimal noisy non-uniform generation algorithm.

Consider arranging the languages in $\mathcal{C}$ in a two-dimensional grid, rows indexed by the noise level $n \geq 0$, and columns by the index $i \geq 1$ of the language in $\mathcal{C}$. For each row n, the element in the i^{th} column, denoted $L_{n,i}$, is simply a copy of L_i. Rows correspond to the different noise levels for each language. We perform a diagonal traversal (Figure 1) of the two-dimensional grid: let $\text{diag}(\mathcal{C}_{n,i})$ denote the sequence of languages in a diagonal traversal starting at $L_{0,1}$ and reaching $L_{n,i}$. That is, $\text{diag}(\mathcal{C}_{n,i}) = (L_{0,1}, L_{1,1}, L_{0,2}, L_{2,1}, L_{1,2}, L_{0,3}, L_{3,1}, L_{2,2}, L_{1,3}, L_{0,4}, \ldots, L_{n,i})$.

For every $i \geq 1$ and $n \geq 0$, we will compute a noisy non-uniform complexity $m^\star_n(L_i)$ in Procedure 2. Our approach will be similar to the noise-free setting from above, where we will perform insertion sort on our diagonal traversal. That is, as we traverse, we will keep track of

Procedure for Computing Noisy Non-uniform Complexities $m^\star_n(L_i)$

1. Initialize $\mathcal{C}'_0 = (), l = 1$.
2. For $n' = 0, 1, 2, \ldots$
   (a) For $h = 0, \ldots, n'$
      i. Set noise level $n = n' - h$, language index $i = h + 1$.
      ii. Initialize $\mathcal{C}'_l$ by appending $L_{n,i}$ at the end of $\mathcal{C}'_{l-1}$, and denote the resulting sequence $\mathcal{C}'_l$ as $(L'_1, \ldots, L'_l)$. This is simply a permutation of $\mathrm{diag}(\mathcal{C}_{n,i})$. We want to run one iteration of "insertion sort" so that $L_{n,i}$ is at its correct position in $\mathcal{C}'_l$. Initialize $j = l$.
      iii. While true do:
         A. Let T be the largest *finite* set such that there exists a subcollection $\mathcal{C}_{\mathrm{check}} \subseteq (L'_1, \ldots, L'_j)$ satisfying the following conditions:
            - $\mathcal{C}_{\mathrm{check}} \ni L'_j$.
            - For every $L' \in \mathcal{C}_{\mathrm{check}}$, which corresponds to some $L_{a,b}$, T is a-contained in L_b.
            - The intersection of languages in $\mathcal{C}_{\mathrm{check}}$ is finite.

            Let $m_{\mathrm{check}} = |T|$.
         B. Suppose L'_{j-1} corresponds to $L_{a,b}$. If $j \leq 1$ or $m_{\mathrm{check}} > m^\star_a(L_b)$, break.
         C. Swap L'_j and L'_{j-1} in $\mathcal{C}'_l$.
         D. $j \leftarrow j - 1$.
      iv. Set $T(L_{n,i}) = T, \mathcal{C}(L_{n,i}) = \mathcal{C}_{\mathrm{check}}, m^\star_n(L_i) = m_{\mathrm{check}}$.
      v. $l \leftarrow l + 1$.

Procedure 2: Insertion Sort for Noisy Non-uniform Generation

an ordering of $\mathrm{diag}(\mathcal{C}_{n,i})$, which we will denote as $\mathcal{C}'_l$. In each iteration of the traversal, we will determine $m^\star_n(L_i)$, and also a subcollection $\mathcal{C}(L_i)$ and set $T(L_i)$ which together act as "witnesses" for $m^\star_n(L_i)$. Our algorithm for noisy non-uniform generation is also based on this diagonal traversal.

Algorithm. Fix a non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ which satisfies $\lim_{t\to\infty} f(t) = \infty$. At time step t, the algorithm follows the procedure of diagonal traversal described through $l = f(t)$ for the purpose of constructing $\mathcal{C}'_{f(t)} = (L'_1, \ldots, L'_{f(t)})$. It then initializes $I_t = ()$, and traverses $L'_1, L'_2, \ldots, L'_{f(t)}$ in this order. Whenever it encounters a language L'_j—which corresponds to some $L_{a,b}$—, it checks if S_t is a-contained in L_b, and that $|\cap_{L\in I_t\cup\{L'_j\}} L| = \infty$. If so, it appends L'_j to I_t. Otherwise, it moves on, leaving I_t unaffected. Observe that at the end of the traversal, I_t is maintained to be a collection of languages that has infinite intersection (if it is non-empty). So, at the end of its traversal, if I_t is empty, the algorithm generates an arbitrary new string from the universe U, and if I_t is non-empty, the algorithm generates a new string from $\cap_{L\in I_t} L$.

Using arguments similar to Section 3, we show that the above algorithm is almost Pareto-optimal.

Theorem 15 (Noisy Generation times) *The algorithm described for noisy generation noisily non-uniformly generates from the collection $\mathcal{C} = (L_1, L_2, \dots,)$, with $t^\star_n(L_i) = \max(g(n,i), m^\star_n(L_i)+1)$, where $g(n,i)$ is the smallest number j that satisfies $f(j) \geq |\mathrm{diag}(\mathcal{C}_{n,i})|$.*

Notice again that by choosing f to be arbitrarily fast-growing, we can have the $\max$ equate to $m^\star_n(L_i) + 1$ for an arbitrarily large prefix. We defer the details, and also the specification of the sufficient condition which guarantees exact Pareto-optimality for the algorithm, to Section B.

5. Pareto-optimal Representative Non-uniform Generation

In representative generation (Peale et al., 2025), we are given a finite partition[5] $\mathcal{A} = \{A_g\}_{g\in[K]}$ of the universe U into K groups. As before, an adversary chooses and enumerates the target language L in a worst-case fashion. But now, the generating algorithm $\mathcal{G}$ outputs a *distribution* $\mathcal{G}_t$ over strings at every time step. Letting S_t denote the set of distinct strings in $x_1, \dots, x_t$, we want the distribution over *groups* that $\mathcal{G}_t$ induces to match the *empirical* distribution over groups seen so far. That is, let Emp_t denote the empirical distribution over U induced by S_t, i.e.,

$$\mathrm{Emp}_t(x) = \frac{\mathbb{1}[x \in S_t]}{|S_t|} \quad \text{for } x \in U. \tag{4}$$

We slightly abuse notation, and use Emp_t and Emp_{S_t} to refer to the same object—this will be convenient when referring to the empirical distribution induced more generally by a given set. For a distribution D over U, let $D^{\mathcal{A}}$ be the distribution it induces over groups. That is, for $g \in [K]$,

$$D^{\mathcal{A}}(g) = \Pr_{x\sim D}[x \in A_g]. \tag{5}$$

Definition 16 (α-representation) *For any accuracy parameter $\alpha \in [0,1]$, we say that the generator $\mathcal{G}$ is α-representative at time step t with respect to the input S_t if $\|\mathrm{Emp}_t^{\mathcal{A}} - \mathcal{G}_t^{\mathcal{A}}\|_\infty \leq \alpha$.*

Definition 17 (Representative Non-uniform Generation (Peale, Raman, and Reingold, 2025))) *Fix an accuracy parameter $\alpha \in [0,1]$. A generating algorithm $\mathcal{G}$ generates non-uniformly in the limit with α-representation from a collection $\mathcal{C} = (L_1, L_2, \dots)$ if for every $L \in \mathcal{C}$, there exists a $t_\alpha(L)$ such that for any enumeration of L presented to the generator, the output distribution $\mathcal{G}_t$ satisfies that $\Pr_{x\sim\mathcal{G}_t}[x \in L \setminus S_t] = 1$ for every t satisfying $|S_t| \geq t_\alpha(L)$, and furthermore, the output group distribution $\mathcal{G}_t^{\mathcal{A}}$ induced by the generator is α-representative with respect to S_t at every $t \geq 1$.*

For representative generation as well, Peale et al. (2025) obtain numerous characterizations; our focus will be on Pareto-optimality. We require the following definition:

Definition 18 (Scarce Groups) *Let $\mathcal{C}$ be a collection of languages, $\mathcal{A} = \{A_g\}_{g\in[K]}$ be a finite partition of the universe U into K groups, and T be a finite subset of U. The scarce groups in $\mathcal{A}$ with respect to $\mathcal{C}$ and T correspond to the set $B := \{g \in [K] : A_g \cap (\cap_{L\in\mathcal{C}} L \setminus T) = \emptyset\}$. We say that T suffers group scarcity (with respect to $\mathcal{C}$ and $\mathcal{A}$), if either there is a scarce group $g \in B$ that satisfies $\mathrm{Emp}_T^{\mathcal{A}}(g) > \alpha$, or it is the case that $\sum_{g\in B} \mathrm{Emp}_T^{\mathcal{A}}(g) > \alpha \cdot (K - |B|)$.*

5. See Lemma 2 in Peale et al. (2025) for pathologies arising with infinitely many groups.

Procedure for Computing Representative Non-uniform Complexities $m^\star_\alpha(L_i)$

1. Initialize $\mathcal{C}'_0 = \emptyset$.
2. For $i = 1, 2, 3, \ldots$
   (a) Initialize $\mathcal{C}'_i$ by appending L_i at the end of $\mathcal{C}'_{i-1}$, and denote the resulting sequence $\mathcal{C}'_i$ as $(L'_1, \ldots, L'_i)$. Initialize $j = i$.
   (b) While true do:
      i. Let T be the largest *finite* set such that there exists a subcollection $\mathcal{C}_{\mathrm{check}} \subseteq (L'_1, \ldots, L'_j)$ satisfying the following conditions:
         A. $\mathcal{C}_{\mathrm{check}} \ni L'_j$.
         B. Every $L \in \mathcal{C}_{\mathrm{check}}$ satisfies that $L \supseteq T$.
         C. T suffers group scarcity with respect to $\mathcal{C}_{\mathrm{check}}$ and $\mathcal{A}$.
         Let $m_{\mathrm{check}} = |T|$.
      ii. If $j \leq 1$ or $m_{\mathrm{check}} > m^\star_\alpha(L'_{j-1})$, break.
      iii. Swap L'_j and L'_{j-1} in $\mathcal{C}'_i$.
      iv. $j \leftarrow j - 1$.
   (c) Set $T(L_i) = T, \mathcal{C}(L_i) = \mathcal{C}_{\mathrm{check}}, m^\star_\alpha(L_i) = m_{\mathrm{check}}$.

Procedure 3: Insertion Sort for Representative Non-uniform Generation

At a high level, the set of scarce groups are precisely those groups, for which we are out of "safe" new strings to generate, assuming the target language is one of the languages in $\mathcal{C}$ that contains T. We will compute the optimal α-representative non-uniform complexity $m^\star_\alpha(L_i)$ for every L_i in Procedure 3 using a similar insertion sort procedure as before, with an appropriately chosen comparison criterion. The procedure motivates the following algorithm, which is qualitatively quite different from the previous two algorithms, requiring a more careful distribution of the generator's probability mass amongst strings from different groups.

Algorithm. Fix a non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ that satisfies $\lim_{t\to\infty} f(t) = \infty$. At time step t, the algorithm follows Procedure 3 up until $i = f(t)$ for the purpose of constructing the ordered collection $\mathcal{C}'_{f(t)} = (L'_1, \ldots, L'_{f(t)})$. It then initializes $I_t = ()$, and traverses $L'_1, L'_2, \ldots, L'_{f(t)}$ in this order. Whenever it comes across a language L'_j that satisfies $L'_j \supseteq S_t$, it checks if S_t *does not* suffer group scarcity with respect to $I_t \cup \{L'_j\}$ and $\mathcal{A}$. Namely, for the set B of scarce groups in $\mathcal{A}$ with respect to $I_t \cup \{L'_j\}$ and S_t, it checks that *every* scarce group $g \in B$ satisfies $\mathrm{Emp}^{\mathcal{A}}_t(g) \leq \alpha$, *and* that $\sum_{g\in B} \mathrm{Emp}^{\mathcal{A}}_t(g) \leq \alpha \cdot (K - |B|)$. If L'_j passes this check, it gets appended to I_t.

At the end of its traversal, if the algorithm finds that I_t is empty, then it sets $\mathcal{G}_t = \mathrm{Emp}_t$. On the other hand, if I_t is non-empty, then consider the scarce groups B with respect to I_t and S_t. Observe first that $B \subsetneq [K]$, since when the last language was inserted into I_t, it was found that $\sum_{g\in B} \mathrm{Emp}^{\mathcal{A}}_t(g) \leq \alpha \cdot (K - |B|)$; if $B = [K]$, the RHS would be 0, but the LHS is positive.

Now, for every non-scarce group $g \in [K] \setminus B$, there exists some string $s_g \in A_g \cap (\cap_{L \in I_t} L \setminus S_t)$; let us arbitrarily pick such an s_g. We will compute the mass that Emp_t assigns to the group g, and assign all of this mass to s_g. That is, $\mathcal{G}_t(s_g) = \mathrm{Emp}_t^{\mathcal{A}}(g)$, and $\mathcal{G}_t(s') = 0$ for every $s' \in A_g$, $s' \neq s_g$. After we do this for all of the non-scarce groups, $\mathcal{G}_t$ will have assigned a total probability mass of $\sum_{g \in [K] \setminus B} \mathrm{Emp}_t^{\mathcal{A}}(g)$, and it still needs to assign a mass worth $\sum_{g \in B} \mathrm{Emp}_t^{\mathcal{A}}(g)$. $\mathcal{G}_t$ will simply distribute this remaining mass evenly over all the strings s_g that were chosen from the non-scarce groups $g \in [K] \setminus B$. Summarily, $\mathcal{G}_t$ has non-zero mass on at most $K - |B|$ strings; each of these strings belong to *all* the languages in I_t, none of these strings belong to S_t, and none of these strings belong to a scarce group. This completes the specification of $\mathcal{G}_t$.

For the algorithm specified above, we can show the following almost Pareto-optimal guarantee with the freedom to choose f as suitable; see Section C for details, and also for a sufficient condition that guarantees exact Pareto-optimality. The proof for α-representation crucially uses the manner in which we spread out the generator's mass in the algorithm, together with the fact that we maintain the set of input strings S_t to *not* suffer group scarcity with respect to the collection I_t and $\mathcal{A}$.

Theorem 19 (Representative Generation times) *The algorithm described for representative generation generates non-uniformly with α-representation from the collection $\mathcal{C} = (L_1, L_2, \ldots)$, with $t^\star_\alpha(L_i) = \max(g(i), m^\star_\alpha(L_i) + 1)$, where $g(i)$ is the smallest number j for which $f(j) \geq i$.*

Acknowledgments

This work was supported by Moses Charikar's and Gregory Valiant's Simons Investigator Awards, and a Google PhD Fellowship.

References

Dana Angluin. Finding patterns common to a set of strings. In *Proceedings of the eleventh annual ACM Symposium on Theory of Computing*, pages 130–141, 1979.

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

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

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

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

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

Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. Characterizations of language generation with breadth. *arXiv preprint arXiv:2412.18530*, 2024. URL https://arxiv.org/abs/2412.18530.

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*, 2025. URL https://arxiv.org/abs/2411.09642.

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

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

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

Charlotte Peale, Vinod Raman, and Omer Reingold. Representative language generation. In *Forty-Second International Conference on Machine Learning*, 2025.

Ananth Raman and Vinod Raman. Generation from noisy examples. In *Forty-Second International Conference on Machine Learning*, 2025.

Appendix A. Impossibility of Pareto-optimality for a Collection That Can Be Uniformly Generated

Proposition 13 (Uniformly Generatable, No Pareto-optimality) *No algorithm can achieve Pareto-optimal non-uniform generation times for the collection $\mathcal{C}$ comprising of languages $\{1, 2, \ldots, 100\} \cup \mathbb{Z}_{\geq i}$ for all $i \in \mathbb{Z}$, where $\mathbb{Z}_{\geq i} = \{i, i+1, i+2, \ldots\}$.*

Proof We first note that the collection can be uniformly generated. To see this, observe that any finite set of size larger than 100 contains a number outside $\{1, 2, \ldots, 100\}$—let i be the smallest such number in the set. Then, the languages that contain the set are precisely the languages $\{1, 2, \ldots, 100\} \cup \mathbb{Z}_{\geq j}$ where $j \leq i$; in particular, the intersection of these languages contains all the numbers that are at least j, and is hence infinite. Thus, the closure dimension of the collection is at most 100. In fact, it is equal to 100, since the intersection of all the languages in the collection is precisely $\{1, 2, \ldots, 100\}$.

Let $L_{e_1}, L_{e_2}, L_{e_3}, \ldots$ be an enumeration of all the languages in $\mathcal{C}$ (with no repetition), where $L_{e_i} = \{1, 2, \ldots, 100\} \cup \mathbb{Z}_{\geq e_i}$ for $e_i \in \mathbb{Z}$. Let $t_1, t_2, t_3, \ldots$ be a sequence of candidate generation times for these languages, where every t_i is finite and at least 1. We call the sequence *admissible* if it satisfies the following property: for every $t \in \{1, 2, \ldots, 100\}$, there exists $n_t < \infty$ such that the set $\mathcal{C}_t = \{L_{e_i} : t_i \leq t\}$ satisfies $e_i \leq n_t$ for every $L_{e_i} \in \mathcal{C}_t$. We claim that there exists an algorithm that achieves the generation times given by the sequence $t_1, t_2, \ldots$ if and only if it is admissible.

For one direction, suppose that the sequence is not admissible, meaning that there exists some $t \in \{1, 2, \ldots, 100\}$ for which the set $\mathcal{C}_t = \{L_{e_i} : t_i \leq t\}$ comprises of languages $L_{e_{i_1}}, L_{e_{i_2}}, \ldots$ where $\lim_{n\to\infty} e_{i_n} = \infty$. Assume for the sake of contradiction that there exists an algorithm that correctly achieves the generation times given in the sequence. Consider feeding $1, 2, \ldots, t$ as input to the algorithm in the first t time steps, and let z_t be the most recent string generated by the algorithm. Then, by assumption, there exists a language $L_{e_n} \in \mathcal{C}_t$ for which $e_n > z_t$. We can now

complete the input by enumerating the rest of the strings in L_{e_n}. Note that the algorithm generated a string outside L_{e_n} at time step t, violating its claimed generation time of $t_i \leq t$ for L_{e_n}.

For the other direction, suppose that the sequence is admissible. Consider the algorithm that operates as follows: at any time step t, if $|S_t| \in \{1, 2 \ldots, 100\}$, it obtains the subcollection $\mathcal{C}_t = \{L_{e_i} : t_i \leq |S_t|\}$; otherwise if $|S_t| > 100$, the algorithm obtains $\mathcal{C}_t = \cap_{L \in \mathcal{C}, S_t \subseteq L} L$. If $\mathcal{C}_t$ is non-empty, and $\cap_{L \in \mathcal{C}_t} L$ is also non-empty, the algorithm generates a new string from $\cap_{L \in \mathcal{C}_t} L$ (or repeats a string if there are no new strings available in this set); otherwise, it generates an arbitrary string. Note that this algorithm is oblivious to its input.

We claim that for every language $L_{e_i} \in \mathcal{C}$, this algorithm achieves the desired generation time of t_i. To see this, it suffices to argue that for every t that satisfies $|S_t| \geq t_i$, $\mathcal{C}_t$ contains L_{e_i}, and that $\cap_{L \in \mathcal{C}_t} L$ is infinite.

First, consider any t for which $t_i \leq |S_t| \leq 100$. By definition of the algorithm then, the subcollection $\mathcal{C}_t$ constructed by the algorithm contains L_{e_i}; in particular, it is non-empty. Furthermore, by admissibility, we have that there exists $n_t < \infty$ such that $e_j \leq n_t$ for every $L_{e_j} \in \mathcal{C}_t$. In particular, this implies that every language in $\mathcal{C}_t$ contains all the numbers that are at least n_t, meaning that $\cap_{L \in \mathcal{C}_t} L$ is infinite.

Now, consider a t for which $|S_t| > 100$; in this case, the algorithm constructs the subcollection $\mathcal{C}_t = \cap_{L \in \mathcal{C}, S_t \subseteq L} L$. Since we are considering L_{e_i} to be the language being enumerated as input, $S_t \subseteq L_{e_i}$. Furthermore, since we argued at the beginning that the closure dimension of $\mathcal{C}$ is 100, we immediately have that $\cap_{L \in \mathcal{C}, S_t \subseteq L} L$ is infinite.

To finish the proof, assume for the sake of contradiction that there exists an algorithm that achieves non-uniform generation times $t_1, t_2, \ldots$ for $L_{e_1}, L_{e_2}, \ldots$ which are Pareto-optimal. By our reasoning above, the sequence $t_1, t_2, \ldots$ must be admissible. But now, consider any $t_i > 1$ in the sequence; such a t_i must exist, since otherwise, if $t_i = 1$ for every i, we can simply input, say 1, at the first time step, post which the algorithm must generate $z \neq 1$, and then the algorithm would have violated its guarantee that $t_i = 1$ for $e_i = z$, if we were to complete enumerating L_{e_i} beyond the first time step. So, for such a t_i that satisfies $t_i > 1$, consider updating $t_i = 1$, and let the other generation times in the sequence remain as is. This updated sequence strictly Pareto-dominates the earlier sequence. Recall also that t_i corresponds to the generation time for the language L_{e_i}; hence, we observe that the updated sequence remains admissible. Namely, each previous value of n_t for $t \in \{1, 2, \ldots, 100\}$ may simply be updated to $n_t = \max(n_t, e_i) < \infty$. Therefore, since the updated sequence is admissible, there exists an algorithm that achieves the generation times given in the updated sequence. Consequently, the algorithm we started with could not have been Pareto-optimal. ■

We conclude by arguing that both the collections from Proposition 12 and Proposition 13 do not satisfy the sufficient condition of Pareto-optimality in Theorem 11. To see this, for either of these collections $\mathcal{C}$, note that the intersection of languages in any finite subcollection is infinite; since the $m^\star(\cdot)$ values in the Pareto-optimal sequence computed in Section 3.1 correspond to finite intersections of finite subcollections, we have that $m^\star(L_j) = 0$ for every $L_j \in \mathcal{C}$ (in fact, this holds true irrespective of the original ordering of $\mathcal{C}$ that we start with to compute the $m^\star(\cdot)$ values). Thus, the condition does not hold for $t = 1$, since $|\{j : m^\star(L_j) + 1 \leq 1\}| = \infty$.

Appendix B. Pareto-optimal Noisy Non-uniform Generation

We first claim that the witness set T constructed in Step A of the while loop in Procedure 2 is always bounded.

Claim 20 (Witness set is bounded) *The set T in Step A of the while loop always has bounded size. In particular, $m^\star_n(L_i) < \infty$ for every n, i.*

Proof Consider any subcollection $\mathcal{C}_{\text{check}}$ of $(L'_1, \ldots, L'_j)$ that contains L'_j, and has finite intersection. In particular, consider the size of the largest finite intersection, i.e.,

$$d^\star = \max_{\substack{\mathcal{C}_{\text{check}} \subseteq (L'_1, \ldots, L'_j), \\ \mathcal{C}_{\text{check}} \ni L'_j, \\ |\cap_{L' \in \mathcal{C}_{\text{check}}} L'| < \infty}} |\cap_{L' \in \mathcal{C}_{\text{check}}} L'|. \tag{6}$$

Since $(L'_1, \ldots, L'_j)$ is finite, there are only finitely many $\mathcal{C}_{\text{check}}$ to consider, and each of these realize a finite intersection. Consequently, $d^\star$ is finite.

Now, let $a^\star$ be the *largest* noise level under consideration in the collection $\text{diag}(\mathcal{C}_{n,i})$, i.e.,

$$a^\star = \max\{a : L_{a,b} \in \text{diag}(\mathcal{C}_{n,i})\},$$

and observe that $a^\star$ is finite (since n', h are finite). Furthermore, since every $L' \in \mathcal{C}_{\text{check}}$ corresponds to some $L_{a,b} \in \text{diag}(\mathcal{C}_{n,i})$, we have that for every $L_{a,b}$ corresponding to $L' \in \mathcal{C}_{\text{check}}$, it is the case that $a \leq a^\star$.

So, suppose for the sake of contradiction that there exists arbitrarily large T, for which there exists a subcollection $\mathcal{C}_{\text{check}}$ satisfying the three conditions listed under Step A. In particular, this means that there exists such a T of size $d > d^\star + l \cdot a^\star$ (where, recall that l is the common size of $\mathcal{C}'_l$ and $\text{diag}(\mathcal{C}_{n,i})$, and an upper bound on the size of $\mathcal{C}_{\text{check}}$). Since T satisfies the second condition in Step A, we have that for every $L_{a,b}$ corresponding to $L' \in \mathcal{C}_{\text{check}}$, T is a-contained in L_b, i.e., $L_{a,b}$ contains at least $|T| - a$ strings in T. But this means that there are at least $|T| - |\mathcal{C}_{\text{check}}| \cdot \max\{a\} \geq d - l \cdot a^\star > d^\star$ strings that are contained in *every* language in $\mathcal{C}_{\text{check}}$. Additionally, $\mathcal{C}_{\text{check}}$ also satisfies the first and third conditions in Step A; that is, $L'_j \in \mathcal{C}_{\text{check}}$, and the intersection of languages in $\mathcal{C}_{\text{check}}$ is finite. Summarily, we have found $\mathcal{C}_{\text{check}}$ to be a subcollection of $(L'_1, \ldots, L'_j)$ that contains L'_j, *and* has a finite intersection of size strictly larger than $d^\star$. This contradicts our derivation of $d^\star$ in (6). Thus, T cannot be unbounded. ■

Next, we show that the sequence of $m^\star_n(L_i)$ values constructed by the procedure forms a sequence of Pareto-optimal generation times. The claim is similar to Claim 6.

Claim 21 ($m^\star_n(\cdot)$ forms a Pareto-optimal sequence) *Any algorithm $\mathcal{G}$ that satisfies $t_{n,\mathcal{G}}(L_i) < m^\star_n(L_i) + 1$ for some L_i must satisfy that $t_{n',\mathcal{G}}(L_j) > m^\star_{n'}(L_j) + 1$ for some $L_{n',j}$ which is before $L_{n,i}$ in $\text{diag}(\mathcal{C}_{n,i})$.*

Proof Note that the language L_i must be such that $m^\star_n(L_i) > 0$, since $t_{n,\mathcal{G}}(\cdot)$ is always at least 1. In particular, this means that $\mathcal{C}(L_{n,i})$ is non-empty. $\mathcal{C}(L_{n,i})$ cannot also only contain copies of L_i, since otherwise, the intersection of languages in $\mathcal{C}(L_{n,i})$ would be infinite. So, $\mathcal{C}(L_{n,i})$ contains at least one language which is not equal to L_i.

Then, the adversary operates as follows: first, the adversary enumerates all the $m^\star_n(L_i)$ strings in $T(L_{n,i})$. Thereafter, the adversary enumerates all the remaining strings in the set $\left(\cap_{L\in\mathcal{C}(L_{n,i})}L\right) \setminus T(L_{n,i})$. Observe that at this point, the adversary has enumerated a sequence $S_t = \{x_1, \ldots, x_t\}$ where $t \geq m^\star_n(L_i)$, such that for every language $L \in \mathcal{C}(L_{n,i})$, which corresponds to some $L_{n',j}$, it is the case that S_t is n'-contained in L_j. Let z be the string generated by $\mathcal{G}$ at time t. Since $\mathcal{G}$ satisfies the guarantee that $t_{n,\mathcal{G}}(L_i) \leq m^\star_n(L_i)$, it must be the case that $z \in L_i \setminus S_t$ (otherwise, the adversary can continue enumerating the rest of L_i beyond this point, producing a valid enumeration of L_i at noise level n, and causing $\mathcal{G}$ to violate its guarantee for L_i). But since we have exhausted out all the strings in the set $\cap_{L\in\mathcal{C}(L_{n,i})}L$, it also must be the case that $z \notin L_{n',j}$ for some $L_{n',j} \in \mathcal{C}(L_{n,i})$. The adversary can then continue enumerating the rest of $L_{n',j}$ beyond t, producing a valid enumeration of L_j at noise level n'. Now, by construction of $\mathcal{C}(L_{n,i})$ and an argument similar to Observation 5, $m^\star_{n'}(L_j) < m^\star_n(L_i)$. So, if $t_{n',\mathcal{G}}(L_j)$ were to be at most $m^\star_{n'}(L_j) + 1$, which is at most $m^\star_n(L_i)$, at time step t, the string generated by $\mathcal{G}$ should have belonged to L_j; we know this is not true, and hence $t_{n',\mathcal{G}}(L_j) > m^\star_{n'}(L_j) + 1$. ■

We now prove a claim similar to Claim 7, which lends itself to the correctness of the stated noisy non-uniform generation algorithm.

Claim 22 (Arg max maintained) *For any $n \geq 0, i \geq 1$, consider any iteration l of the nested for loop after the diagonal traversal has reached $L_{n,i}$ (i.e., $l \geq |\mathrm{diag}(\mathcal{C}_{n,i})|$), and let k be the index of $L_{n,i}$ in $\mathcal{C}'_l = (L'_1, \ldots, L'_l)$ at the end of this iteration. Then,*

$$T(L_{n,i}) \in \arg\max_T \{|T| : T \textit{ satisfies } (\star)\}, \tag{7}$$

where $(\star)$ is the following condition: T is finite, and there exists a subcollection $\mathcal{C}_{\mathrm{check}} \subseteq (L'_1, \ldots, L'_k)$, such that $\mathcal{C}_{\mathrm{check}} \ni L'_k$, the intersection of languages in $\mathcal{C}_{\mathrm{check}}$ is finite, and for every $L' \in \mathcal{C}_{\mathrm{check}}$, which corresponds to some $L_{a,b}$, T is a-contained in L_b.

Proof At the end of the iteration l of the nested for loop when the diagonal traversal is *at* $L_{n,i}$ (i.e., $l = |\mathrm{diag}(\mathcal{C}_{n,i})|$), the claim is true by definition of $T(L_{n,i})$. Now suppose as our inductive hypothesis that the claim is true for some iteration $l' \geq l$. We will show that it is true for $l' + 1$. Suppose k was the index of $L_{n,i}$ at the end of the l'^{th} iteration of the nested for loop. If k continues to be the index of $L_{n,i}$ at the end of the $(l'+1)^{\text{th}}$ iteration as well (meaning that the language after $L_{n,i}$ in the diagonal traversal—call it L_{next}—was placed at some index larger than k in $\mathcal{C}'_{l'+1}$), the claim continues to be true simply by the inductive hypothesis. Otherwise, L_{next} was placed at an index in $\{1, \ldots, k\}$, and so $L_{n,i}$ was moved to index $k+1$ in $\mathcal{C}'_{l'+1}$. Consider the precise iteration of the while loop when j was equal to $k+1$ (so that L'_{k+1} was L_{next}), and m_{check} was found to be at most $m^\star_n(L_i)$. At this point, Step C tells us to swap L'_k and L'_{k+1} in $\mathcal{C}'_{n+1}$ (so that L'_{k+1} becomes equal to $L_{n,i}$, and L'_k becomes equal to L_{next}), and thereafter, note that the (unordered) subcollection of languages before L'_{k+1} remains unchanged until the while loop breaks, and the iteration of the for loop completes. So, it suffices to argue that, *before swapping*, $T(L_{n,i})$ belongs to the $\arg\max$ of $|T|$ over the set $\{T\}$, where T satisfies the condition: T is finite, and there exists a subcollection $\mathcal{C}_{\mathrm{check}} \subseteq (L'_1, \ldots, L'_k, L'_{k+1})$, such that $\mathcal{C}_{\mathrm{check}} \ni L'_k$, the intersection of languages in $\mathcal{C}_{\mathrm{check}}$ is finite, and for every $L' \in \mathcal{C}_{\mathrm{check}}$, which corresponds to some $L_{a,b}$, T is a-contained in L_b. Recall that our inductive hypothesis already guarantees that $T(L_{n,i})$ belongs to the arg max over the set of $\{T\}$,

where the condition on T only considers subcollections of $(L'_1, \ldots, L'_k)$. That is, the only additional language now being considered in the condition is L'_{k+1}. So, if the max increased from what it was when L'_{k+1} was not being considered, it *must* be the case that for every T belonging to the new arg max, the subcollection $\mathcal{C}_{\text{check}}$ for that T contains L'_{k+1}. But this means that such a T—which has size strictly larger than $T(L_{n,i})$—satisfies the condition: T is finite, and there exists a subcollection $\mathcal{C}_{\text{check}} \subseteq (L'_1, \ldots, L'_k, L'_{k+1})$, such that $\mathcal{C}_{\text{check}} \ni L'_{k+1}$, the intersection of languages in $\mathcal{C}_{\text{check}}$ is finite, and for every $L' \in \mathcal{C}_{\text{check}}$, which corresponds to some $L_{a,b}$, T is a-contained in L_b. This contradicts that m_{check} is at most $m^\star_n(L_i)$ (since $|T(L_{n,i})| = m^\star_n(L_i)$). Thus, it must be the case that $T(L_{n,i})$ continues to be in the arg max. This completes the proof of the inductive step. ■

We can now argue the correctness of the noisy generation algorithm stated in Section 4, showing also that it is almost Pareto-optimal.

Theorem 15 (Noisy Generation times) *The algorithm described for noisy generation noisily non-uniformly generates from the collection $\mathcal{C} = (L_1, L_2, \ldots,)$, with $t^\star_n(L_i) = \max(g(n,i), m^\star_n(L_i) + 1)$, where $g(n,i)$ is the smallest number j that satisfies $f(j) \geq |\text{diag}(\mathcal{C}_{n,i})|$.*

Proof Suppose that the adversary chose a noise level $n \geq 0$, and presented the algorithm with an enumeration of L_i at noise level n, and suppose that we are at a time step t for which $|S_t| \geq t^\star_n(L_i, \mathcal{C})$. Observe that since $t \geq |S_t| \geq g(n,i)$, and f is non-decreasing, $L_{n,i}$ is under consideration at t, and belongs to the collection $\mathcal{C}'_{f(t)} = (L'_1, \ldots, L'_{f(t)})$ constructed by the algorithm. Let k be the index of the language $L_{n,i}$ in this collection. Note that S_t is n-contained in L_i, since the adversary is in fact enumerating L_i at noise level n. So, we only need to argue that when $L'_k = L_{n,i}$ is encountered by the algorithm as it traverses $\mathcal{C}'_{f(t)}$, it finds that $|\cap_{L \in I_t \cup \{L'_k\}} L| = \infty$. Assume for the sake of contradiction that $|\cap_{L \in I_t \cup \{L'_k\}} L| < \infty$. In this case, observe that we have found a finite S_t of size at least $m^\star_n(L_i) + 1$, for which $I_t \cup \{L'_k\}$ is a subcollection of $(L'_1, \ldots, L'_k)$, that contains L'_k, has finite intersection, and for every L in the subcollection, which corresponds to some $L_{a,b}$, S_t is a-contained in L_b. But this implies that the maximum in the RHS of (7) in Claim 22 is at least $m^\star_n(L_i) + 1$, which contradicts that $T(L_{n,i})$ realizes the maximum (since $|T(L_{n,i})| = m^\star_n(L_i)$). Thus, $L_{n,i}$ passes both the checks when it is encountered by the algorithm, and hence gets added to I_t. ■

Finally, we derive a sufficient condition for when the algorithm is exactly Pareto-optimal.

Theorem 23 (Sufficient Condition for Exact Pareto-optimality) *Let $\mathcal{C} = (L_1, L_2, \ldots)$ be a collection that satisfies the following property: for every $t \in \mathbb{N}$, $|\{(n', j) : m^\star_{n'}(L_j) + 1 \leq t\}| < \infty$, where the $m^\star_n(\cdot)$ values are those computed in Procedure 2. Then, there exists a noisy non-uniform generation algorithm that achieves a Pareto-optimal sequence of generation times for $\mathcal{C}$.*

Proof Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined as $f(t) = \max_{n',j}\{|\text{diag}(\mathcal{C}_{n',j})| : m^\star_{n'}(L_j) + 1 \leq t\}$. Observe that f is a non-decreasing function, and that $\lim_{t\to\infty} f(t) = \infty$. Furthermore, observe also that for any n, i, $f(m^\star_n(L_i) + 1) \geq |\text{diag}(\mathcal{C}_{n,i})|$. Therefore, if $g(n,i)$ is the smallest number j for which $f(j) \geq |\text{diag}(\mathcal{C}_{n,i})|$, then $g(n,i) \leq m^\star_n(L_i) + 1$. So, if we choose to run the algorithm given in Section 4 with this function f, the guarantee of Theorem 15 implies $t^\star_n(L_i) = m^\star_n(L_i) + 1$ for every L_i enumerated at noise level n. From Claim 21, we conclude that the algorithm is Pareto-optimal. ■

Appendix C. Pareto-optimal Representative Non-uniform Generation

We will first show that every $m^\star_\alpha(L_i)$ is finite.

Claim 24 (Witness set is bounded) *The set T in Step i of the while loop in Procedure 3 always has bounded size. In particular, $m^\star_\alpha(L_i) < \infty$ for every L_i.*

Proof Consider any subcollection $\mathcal{C}_{\text{check}}$ of $(L'_1, \ldots, L'_j)$ that contains L'_j. Let $T(\mathcal{C}_{\text{check}})$ be a finite subset of $\cap_{L\in\mathcal{C}_{\text{check}}} L$ which suffers group scarcity with respect to $\mathcal{C}_{\text{check}}$ and $\mathcal{A}$. We will argue that $T(\mathcal{C}_{\text{check}})$ cannot be arbitrarily large. It suffices to reason about the case when $\cap_{L\in\mathcal{C}_{\text{check}}} L$ is infinite, since otherwise, $T(\mathcal{C}_{\text{check}})$ is immediately bounded.

To this end, consider the set of groups R_1 for which

$$R_1 = \{g \in [K] : A_g \cap (\cap_{L\in\mathcal{C}_{\text{check}}} L) = \infty\}. \tag{8}$$

Note that since we assumed $\cap_{L\in\mathcal{C}_{\text{check}}} L$ is infinite, and $\mathcal{A}$ is a partition of U, R_1 is necessarily non-empty. On the other hand, let R_2 denote the groups for which

$$R_2 = \{g \in [K] : A_g \cap (\cap_{L\in\mathcal{C}_{\text{check}}} L) < \infty\}. \tag{9}$$

In particular, let

$$p = \max_{g\in R_2} |A_g \cap (\cap_{L\in\mathcal{C}_{\text{check}}} L)\,|. \tag{10}$$

Using that $K < \infty$, and the definition of the set R_2, we have that $p < \infty$. We can now claim that the size of $T(\mathcal{C}_{\text{check}})$ may be at most $\frac{Kp}{\alpha}$, which is a finite number. To see this, suppose $T(\mathcal{C}_{\text{check}})$ is a finite subset of $\cap_{L\in\mathcal{C}_{\text{check}}} L$ that has size strictly larger than $\frac{Kp}{\alpha}$. Observe that the scarce groups for $T(\mathcal{C}_{\text{check}})$ must necessarily be a subset of R_2. In that case, the largest that $\text{Emp}^{\mathcal{A}}_{T(\mathcal{C}_{\text{check}})}(g)$ can be is if we include all the strings in $A_g \cap (\cap_{L\in\mathcal{C}_{\text{check}}} L)$ in $T(\mathcal{C}_{\text{check}})$—but even in this case, we would have $\text{Emp}^{\mathcal{A}}_{T(\mathcal{C}_{\text{check}})}(g) < \frac{\alpha}{K} \le \alpha$. On the other hand, observe also that

$$\sum_{g\in B} \text{Emp}^{\mathcal{A}}_{T(\mathcal{C}_{\text{check}})}(g) < \frac{\alpha|B|}{K} < \alpha < \alpha(K - |B|).$$

Here, we used that $|B| < K$—this follows because we assumed that $\cap_{L\in\mathcal{C}_{\text{check}}} L$ is infinite, $T(\mathcal{C}_{\text{check}})$ is finite, and that $\mathcal{A}$ is a partition of U, which together imply that not all groups can be scarce. Thus, we can see that $T(\mathcal{C}_{\text{check}})$ would not satisfy group scarcity with respect to $\mathcal{C}_{\text{check}}$ and $\mathcal{A}$, if its size is larger than $\frac{Kp}{\alpha}$.

Summarily, we have shown that for any $\mathcal{C}_{\text{check}}$ that contains L'_j, the largest that a finite subset $T(\mathcal{C}_{\text{check}})$ of $\cap_{L\in\mathcal{C}_{\text{check}}} L$ can be while suffering group scarcity with respect to $\mathcal{C}_{\text{check}}$ and $\mathcal{A}$ is at most $\frac{Kp}{\alpha} := f(\mathcal{C}_{\text{check}}) < \infty$. So, define

$$d^\star = \max_{\substack{\mathcal{C}_{\text{check}}\subseteq(L'_1,\ldots,L'_j)\\ \mathcal{C}_{\text{check}}\ni L'_j}} f(\mathcal{C}_{\text{check}}). \tag{11}$$

Since the collection $(L'_1, \ldots, L'_j)$ is finite, there are only finitely many $\mathcal{C}_{\text{check}}$ to consider, and hence $d^\star$ is finite.

Finally, suppose for the sake of contradiction that there were to exist arbitrarily large finite T that satisfy the conditions in Step i of the while loop. This would mean that there exists some T of finite size $d > d^\star$, and a subcollection $\mathcal{C}_{\text{check}}$ of $(L'_1, \ldots, L'_j)$, such that $\mathcal{C}_{\text{check}}$ contains L'_j, $T \subseteq \cap_{L \in \mathcal{C}_{\text{check}}} L$, and also, T suffers group scarcity with respect to $\mathcal{C}_{\text{check}}$ and $\mathcal{A}$. But this contradicts our derivation of (11) above, and concludes the proof that T cannot be arbitrarily large. ■

We can now argue that the sequence of $m^\star_\alpha(L_i)$ values forms a Pareto-optimal sequence.

Claim 25 ($m^\star_\alpha(\cdot)$ forms a Pareto-optimal sequence) *Any algorithm $\mathcal{G}$ that satisfies $t_{\alpha,\mathcal{G}}(L_i) < m^\star_\alpha(L_i) + 1$ for some L_i must satisfy that $t_{\alpha,\mathcal{G}}(L_j) > m^\star_\alpha(L_j) + 1$ for some $j < i$.*

Proof Note that the language L_i must be such that $m^\star_\alpha(L_i) > 0$, since $t_{\alpha,\mathcal{G}}(\cdot)$ is always at least 1. In particular, this means that $\mathcal{C}(L_i)$ is non-empty.

Suppose an adversary enumerates all the $m^\star_\alpha(L_i)$ strings in $T(L_i)$ in the first $m^\star_\alpha(L_i)$ time steps. Consider the distribution $\mathcal{G}_t$ output by the algorithm $\mathcal{G}$ at time $t = m^\star_\alpha(L_i)$. By assumption, $\mathcal{G}_t$ satisfies the properties required of α-representation at t. There are two cases based on the mass that $\mathcal{G}_t$ assigns to the set $\cap_{L \in \mathcal{C}(L_i)} L \setminus S_t$:

Case 1: $\Pr_{x \sim \mathcal{G}_t}\left[x \in \cap_{L \in \mathcal{C}(L_i)} L \setminus S_t\right] = 1$**.** The adversary can continue the enumeration of any $L \in \mathcal{C}(L_i)$ beyond t. Note that since $S_t = T(L_i)$, by definition of the set $T(L_i)$, we have that S_t suffers group scarcity with respect to $\mathcal{C}(L_i)$ and $\mathcal{A}$. That is, if B denotes the set of scarce groups with respect to $\mathcal{C}(L_i)$ and S_t, then either there exists a scarce group $g \in B$ that satisfies $\text{Emp}^{\mathcal{A}}_t(g) > \alpha$, or it is the case that $\sum_{g \in B} \text{Emp}^{\mathcal{A}}_t(g) > \alpha \cdot (K - |B|)$. Observe that $\mathcal{G}^{\mathcal{A}}_t(g) = \Pr_{x \sim \mathcal{G}_t}[x \in A_g] = 0$ for every $g \in B$; otherwise, since $\mathcal{G}_t$ puts all its mass on $\cap_{L \in \mathcal{C}(L_i)} L \setminus S_t$ in the present case, we would have found a string in some $A_g \cap \left(\cap_{L \in \mathcal{C}(L_i)} L \setminus S_t\right)$, which contradicts that g is a scarce group.

So, suppose that there exists a scarce group $g \in B$ that satisfies $\text{Emp}^{\mathcal{A}}_t(g) > \alpha$. This implies that $|\text{Emp}^{\mathcal{A}}_t(g) - \mathcal{G}^{\mathcal{A}}_t(g)| = \text{Emp}^{\mathcal{A}}_t(g) > \alpha$, which contradicts the α-representation property of $\mathcal{G}$. Thus, we cannot have $\text{Emp}^{\mathcal{A}}_t(g) > \alpha$ for any scarce group g.

Now, suppose that $\sum_{g \in B} \text{Emp}^{\mathcal{A}}_t(g) > \alpha \cdot (K - |B|)$. This would imply that

$$\begin{aligned}
\sum_{g \in [K]} \mathcal{G}^{\mathcal{A}}_t(g) = \sum_{g \in [K] \setminus B} \mathcal{G}_t(g) &\leq \sum_{g \in [K] \setminus B} (\text{Emp}^{\mathcal{A}}_t(g) + \alpha) \\
&= \sum_{g \in [K] \setminus B} \text{Emp}^{\mathcal{A}}_t(g) + \alpha \cdot (K - |B|) \\
&< \sum_{g \in [K] \setminus B} \text{Emp}^{\mathcal{A}}_t(g) + \sum_{g \in B} \text{Emp}^{\mathcal{A}}_t(g) \\
&= \sum_{g \in [K]} \text{Emp}^{\mathcal{A}}_t(g) = 1,
\end{aligned}$$

which contradicts that $\mathcal{G}^{\mathcal{A}}_t$ is a valid probability distribution. Summarily, we deduce that Case 1 cannot happen.

Case 2: $\Pr_{x \sim \mathcal{G}_t}\left[x \in \cap_{L \in \mathcal{C}(L_i)} L \setminus S_t\right] < 1$**.** In this case, observe that there exists $L \in \mathcal{C}(L_i)$, such that $\Pr_{x \sim \mathcal{G}_t}[x \notin L \setminus S_t] > 0$. Moreover, $\mathcal{C}(L_i)$ cannot only contain copies of L_i; otherwise, the adversary can continue enumerating the rest of L_i beyond $t = m^\star_\alpha(L_i)$, and $\mathcal{G}$ will have violated

outputting a valid string at time t. So, it must be the case that $L \neq L_i$. The adversary can then continue enumerating this L beyond t. Observe that by the manner in which $m^\star_\alpha(L_i)$ and $\mathcal{C}(L_i)$ are determined, such an L corresponds to some L_j for $j < i$ in the given specification of the collection $\mathcal{C} = (L_1, L_2, \ldots)$. Furthermore, by an argument identical to Observation 5, it is also the case that $m^\star_\alpha(L_j) < m^\star_\alpha(L_i)$. So, if it were the case that $t_{\alpha,\mathcal{G}}(L_j, \mathcal{C})$ is at most $m^\star_\alpha(L_j) + 1$, which is at most $m^\star_\alpha(L_i)$, then $\mathcal{G}_t$ should satisfy the guarantee that $\Pr_{x\sim\mathcal{G}_t}[x \in L_j \setminus S_t] = 1$; we just argued that this is not true, concluding the proof. ■

The following claim shows again that whenever a language is moved forward when performing the insertion sort, its witness set continues to remain in the arg max of the relevant criterion in the prefix of its new position.

Claim 26 (Arg max maintained) *For every $i \geq 1$, and for every $t \geq i$, suppose k is the index of language L_i in $\mathcal{C}'_t = (L'_1, \ldots, L'_t)$ at the end of the t^{th} iteration of the for loop. Then, $T(L_i)$ satisfies that*

$$T(L_i) \in \arg\max_T \{|T| : T \textit{ satisfies } (\star)\}, \tag{12}$$

where $(\star)$ is the following condition: T is finite, and there exists a subcollection $\mathcal{C}_{\text{check}} \subseteq (L'_1, \ldots, L'_k)$, such that $\mathcal{C}_{\text{check}} \ni L'_k$, every $L \in \mathcal{C}_{\text{check}}$ contains T, and T suffers group scarcity with respect to $\mathcal{C}_{\text{check}}$ and $\mathcal{A}$.

Proof For $t = i$, the claim is true by definition of $T(L_i)$. Now suppose as our inductive hypothesis that the claim is true for some $t \geq i$. We will show that it is true for $t + 1$. Suppose k was the index of L_i at the end of the t^{th} iteration of the for loop. If k continues to be the index of L_i at the end of the $(t+1)^{\text{th}}$ iteration as well (that is, the language L_{t+1} was placed at some index larger than k in $\mathcal{C}'_{t+1}$), the claim continues to be true by the inductive hypothesis. Otherwise, the language L_{t+1} was placed at an index in $\{1, \ldots, k\}$, and so L_i was moved to index $k + 1$ in $\mathcal{C}'_{t+1}$. Consider the precise iteration of the while loop when j was equal to $k + 1$ (so that L'_{k+1} was L_{t+1}), and m_{check} was found to be at most $m^\star_\alpha(L'_k)$. At this point, Step iii would tell us to swap L'_k and L'_{k+1} (so that L'_{k+1} becomes equal to L_i, and L'_k becomes equal to L_{t+1}), and thereafter, observe that the (unordered) subcollection of languages before L'_{k+1} remains unchanged until the while loop breaks, and the iteration of the for loop completes. So, it suffices to argue that, *before swapping*, $T(L_i)$ belongs to the arg max of $|T|$ over the set $\{T\}$, where T satisfies the condition: T is finite, and there exists a subcollection $\mathcal{C}_{\text{check}} \subseteq (L'_1, \ldots, L'_k, L'_{k+1})$ where $\mathcal{C}_{\text{check}} \ni L'_k$, every language $L \in \mathcal{C}_{\text{check}}$ contains T, and furthermore, T suffers group scarcity with respect to $\mathcal{C}_{\text{check}}$ and $\mathcal{A}$. Recall that our inductive hypothesis already guarantees that $T(L_i)$ belongs to the arg max over the set $\{T\}$, where the condition on T only considers subcollections of $(L'_1, \ldots, L'_k)$. That is, the only additional language now being considered in the condition is L'_{k+1}. So, if by the introduction of this new language in the condition, the max were to increase, it must be the case that for every T belonging to the new arg max, the subcollection *necessarily* contains L'_{k+1}. But this means that such a T, which is finite and has size strictly larger than $T(L_i)$, satisfies that: there exists a subcollection $\mathcal{C}_{\text{check}} \subseteq (L'_1, \ldots, L'_{k+1})$, $\mathcal{C}_{\text{check}} \ni L'_{k+1}$, every language $L \in \mathcal{C}_{\text{check}}$ contains T, and furthermore, T also suffers group scarcity with respect to $\mathcal{C}_{\text{check}}$ and $\mathcal{A}$. This contradicts m_{check} being at most $m^\star_\alpha(L'_k)$ (since $m^\star_\alpha(L'_k) = m^\star_\alpha(L_i) = |T(L_i)|$). Thus, it must be the case that $T(L_i)$ continues to be in the arg max, completing the inductive step. ■

Theorem 19 (Representative Generation times) *The algorithm described for representative generation generates non-uniformly with α-representation from the collection $\mathcal{C} = (L_1, L_2, \ldots)$, with $t^\star_\alpha(L_i) = \max(g(i), m^\star_\alpha(L_i) + 1)$, where $g(i)$ is the smallest number j for which $f(j) \geq i$.*

Proof First, observe that for any t satisfying $|S_t| \geq g(i)$, L_i belongs to the collection $\mathcal{C}'_{f(t)} = (L'_1, \ldots, L'_{f(t)})$ under consideration by the algorithm. Suppose k is the index for which $L'_k = L_i$. We only need to argue that L'_k satisfies the required check. Assume for the sake of contradiction that L'_k fails the check. In this case, S_t, which has size at least $m^\star_\alpha(L_i) + 1$, satisfies that $I_t \cup \{L'_k\}$ is a subcollection of $(L'_1, \ldots, L'_k)$ which contains L'_k, all languages in the subcollection contain S_t, and furthermore, S_t suffers group scarcity with respect to $I_t \cup \{L'_k\}$ and $\mathcal{A}$. Thus, the set I_t constructed by the algorithm above contains L_i.

We will now show that, for any enumeration of L_i, the output distribution $\mathcal{G}_t$ satisfies $\Pr_{x\sim\mathcal{G}_t}[x \in L_i \setminus S_t] = 1$ for every t satisfying $|S_t| \geq \max(g(i), m^\star_\alpha(L_i) + 1)$. As argued above, we know that the collection I_t constructed by the algorithm at time t contains L_i; in particular, it is non-empty. We also justified in the description of the algorithm that in the case that I_t is non-empty, $\mathcal{G}_t$ has mass only on strings *not* belonging to S_t, and belonging to *every* language in I_t. Since I_t contains L_i, we are done.

We will now argue that the algorithm satisfies α-representation for every $t \geq 1$. In the case that I_t is found to be empty, observe that $\mathcal{G}_t$ is set to be Emp_t, and so, $\|\mathcal{G}^{\mathcal{A}}_t - \mathrm{Emp}^{\mathcal{A}}_t\|_\infty = 0$.

So, consider the case where I_t is non-empty, and consider the set B of scarce groups with respect to I_t and S_t. Observe that by construction of I_t, S_t does not suffer group scarcity with respect to I_t and $\mathcal{A}$. This means that for every scarce group $g \in B$, $\mathrm{Emp}^{\mathcal{A}}_t(g) \leq \alpha$, and also, $\sum_{g\in B} \mathrm{Emp}^{\mathcal{A}}_t(g) \leq \alpha \cdot (K - |B|)$. Furthermore, when I_t is non-empty, recall that the algorithm puts no mass on any string belonging to a scarce group. Thus, for every scarce group $g \in B$, we have that $|\mathcal{G}^{\mathcal{A}}_t(g) - \mathrm{Emp}^{\mathcal{A}}_t(g)| = |\mathrm{Emp}^{\mathcal{A}}_t(g)| \leq \alpha$.

Now, consider a non-scarce group $g \in [K] \setminus B$. Recall that the mass that the algorithm assigns to this group is entirely concentrated on the single string s_g that it chooses. Furthermore, this mass is exactly equal to $\mathrm{Emp}^{\mathcal{A}}_t(g)$, *plus* an amount that is equal to $\frac{1}{K-|B|}\sum_{g\in B} \mathrm{Emp}^{\mathcal{A}}_t(g)$, where the additional mass is the result of the algorithm distributing the mass that $\mathrm{Emp}^{\mathcal{A}}_t$ has on scarce groups evenly among the non-scarce groups. So, we have that

$$|\mathcal{G}^{\mathcal{A}}_t(g) - \mathrm{Emp}^{\mathcal{A}}_t(g)| = \left|\frac{1}{K-|B|}\sum_{g\in B} \mathrm{Emp}^{\mathcal{A}}_t(g)\right| \leq \frac{\alpha\cdot(K-|B|)}{K-|B|} = \alpha.$$

Thus, we have established that $\|\mathcal{G}^{\mathcal{A}}_t - \mathrm{Emp}^{\mathcal{A}}_t\|_\infty \leq \alpha$. ■

We conclude by stating the corresponding sufficient condition for exact Pareto-optimality.

Theorem 27 (Sufficient Condition for Exact Pareto-optimality) *Let $\mathcal{C} = (L_1, L_2, \ldots)$ be a collection that satisfies the following property: for every $t \in \mathbb{N}$, $|\{j : m^\star_\alpha(L_j) + 1 \leq t\}| < \infty$, where the $m^\star_\alpha(\cdot)$ values are those computed in Procedure 3. Then, there exists an algorithm that achieves a Pareto-optimal sequence of non-uniform generation times for $\mathcal{C}$.*

Proof Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined as $f(t) = \max\{j : m^\star_\alpha(L_j) + 1 \leq t\}$. Observe that f is a non-decreasing function, and that $\lim_{t\to\infty} f(t) = \infty$. Furthermore, observe also that for

any i, $f(m^\star_\alpha(L_i)+1) \geq i$. Therefore, if $g(i)$ is the smallest number j for which $f(j) \geq i$, then $g(i) \leq m^\star_\alpha(L_i)+1$. So, if we choose to run the algorithm given in Section 5 with this function f, the guarantee of Theorem 19 implies $t^\star_\alpha(L_i) = m^\star_\alpha(L_i)+1$ for every L_i. From Claim 25, we conclude that the algorithm is Pareto-optimal. ■