EXPLORING FACETS OF LANGUAGE GENERATION IN THE LIMIT

Chirag Pabbaraju

Stanford University

Joint work with
Moses Charikar

LANGUAGE GENERATION

Given a finite set of training examples from some unknown language, produce new strings from the language that don't already appear in the training data

"Inbox overflows,
each ping a quiet demand —
I mute the world now."

⋮

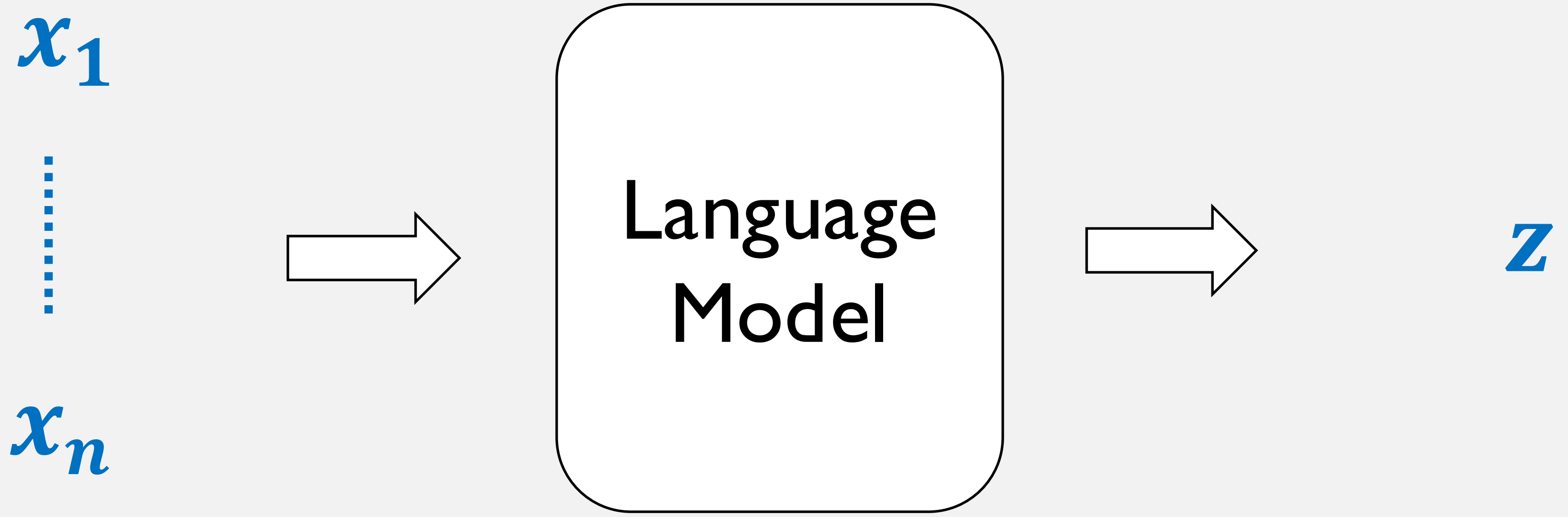
"One text left unread,
not from lack of attention —
I just need some peace."

Language Model

"Cracked phone screen again,
a spiderweb of mistakes —
I still swipe through it."

LANGUAGE GENERATION

Given a finite set of training examples from some unknown language, produce new strings from the language that don't already appear in the training data



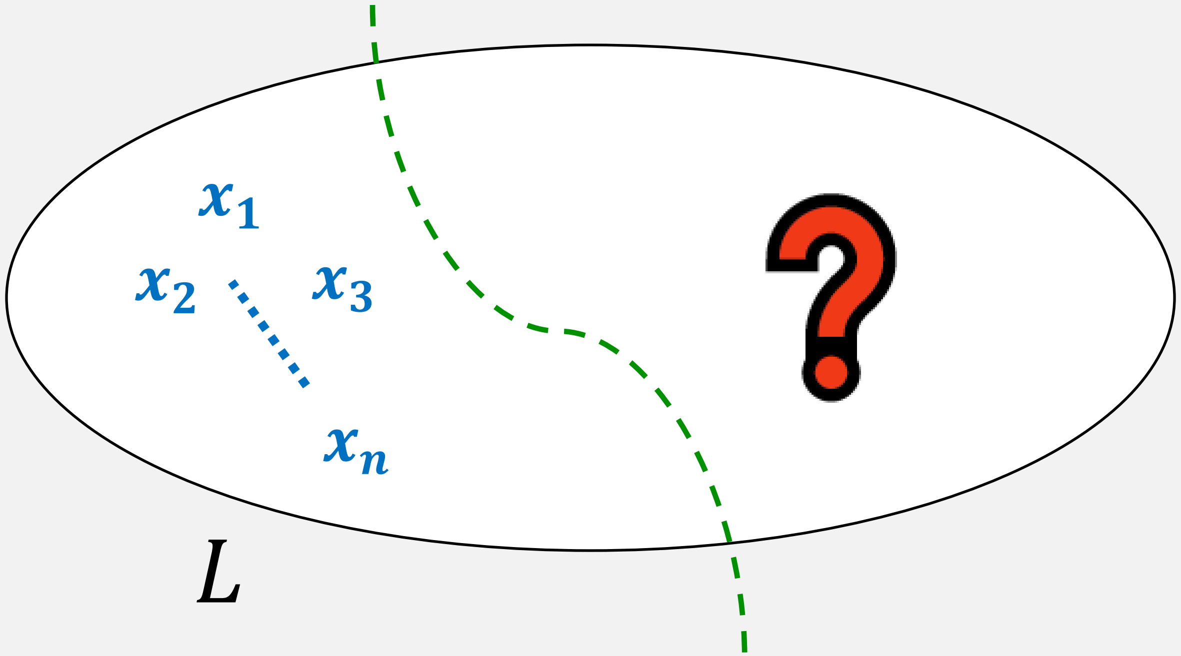
LANGUAGE GENERATION

Given a finite set of training examples from some unknown language, produce new strings from the language that don't already appear in the training data

No structural assumptions

…

Intractable?



LANGUAGE IDENTIFICATION IN THE LIMIT

(Gold '67)

- Known collection $\mathcal{C} = \{L_1, L_2, L_3, \dots\}$
- Adversary chooses some target language $L = L_z$, starts enumerating it in an order of their choosing

$$x_1, x_2, x_3, x_4, x_5, \dots\dots$$

Every $x \in L$ appears at some time, repeats allowed

- At each time step t, algorithm makes a guess of the index of the language that is being enumerated
- Identifies in the limit if beyond some large enough $t^\star$, all guesses correct

chooses L_{70}

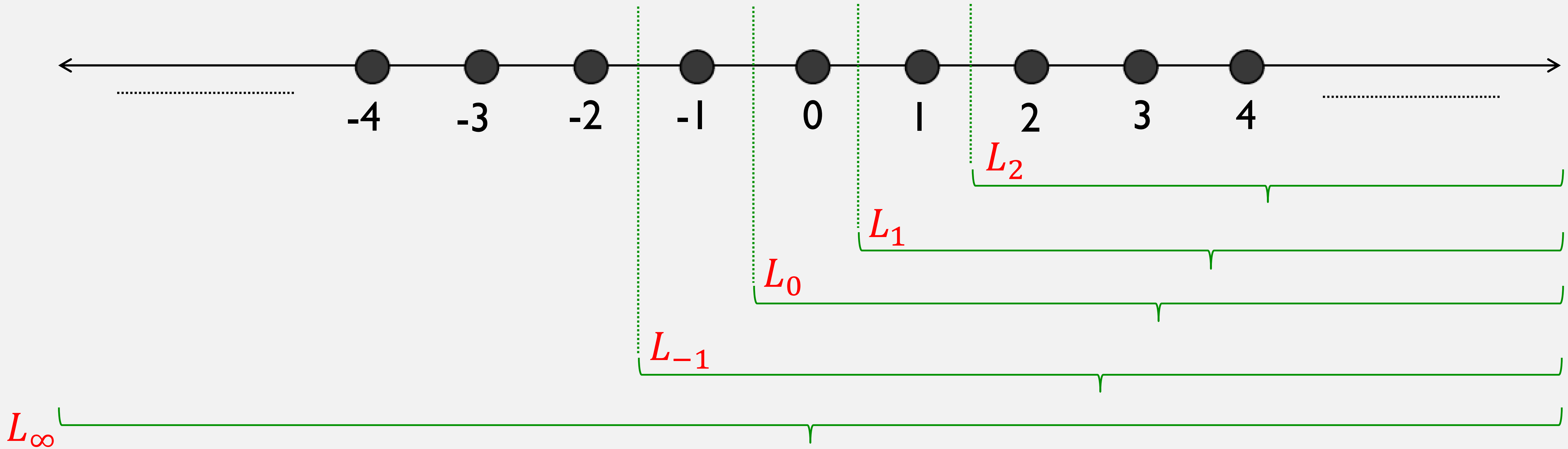
x_1	x_2	x_3	x_4	....	$x_{t^\star}$	$x_{t^\star+1}$ ....
L_2	L_2	L_{10}	L_{21}	....	L_{70}	L_{70} ....

LANGUAGE IDENTIFICATION IN THE LIMIT

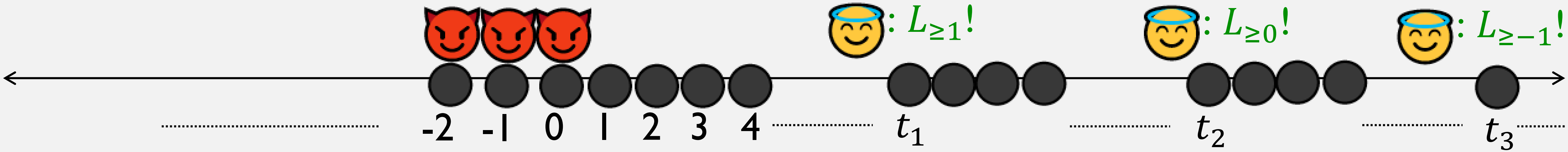
- Example: $\mathcal{C} = \{L_{\text{even integers}}, L_{\text{all integers}}\}$
- Algorithm keeps guessing $L_{\text{even integers}}$ up until the time it sees an odd integer for the first time, at which point it switches to $L_{\text{all integers}}$
- If adversary chose $L_{\text{even integers}}$, algorithm is correct from $t = 1$
- Otherwise, adversary must reveal an odd integer: correct from that point
- Hopelessly hard for essentially any interesting infinite collection (Gold '67) 😔
- Theorem (Angluin '80): Collection identifiable iff it satisfies Angluin's condition…. (very restrictive)

HARD INSTANCE FOR IDENTIFICATION

$$L_\infty = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$
$$L_{\geq i} = \{i, i+1, i+2, i+3, \dots\}$$



HARD INSTANCE FOR IDENTIFICATION



Valid enumeration of $L_{\geq 1}$; at some finite time t_1 (and beyond), algorithm must guess $L_{\geq 1}$

Valid enumeration of $L_{\geq 0}$; at some finite time t_2 (and beyond), algorithm must guess $L_{\geq 0}$

Valid enumeration of $L_{\geq -1}$; at some finite time t_3 (and beyond), algorithm must guess $L_{\geq -1}$

Adversary repeats this game, produces a valid enumeration of L_∞

Infinite sequence $t_1 < t_2 < t_3 < \cdots$ where algorithm makes a mistake 😔

LANGUAGE GENERATION IN THE LIMIT

(Kleinberg-Mullainathan'24)

- $\mathcal{C} = \{L_1, L_2, L_3, \dots\}$
- Adversary chooses some target language L_z, starts enumerating it in an order of their choosing

$$x_1, x_2, x_3, x_4, x_5, \ldots\ldots.$$

- At each time step t, algorithm generates a string z_t
- Generates in the limit if beyond some large enough $t^\star$, all strings generated are new and in L_z

chooses L_{70}

😈	x_1	x_2	x_3	x_4	....	$x_{t^\star}$	$x_{t^\star+1}$	....
😇	z_1	z_2	z_3	z_4	....	$z_{t^\star}$ new, $\in L_{70}$	$z_{t^\star+1}$ new, $\in L_{70}$	....

LANGUAGE GENERATION IN THE LIMIT

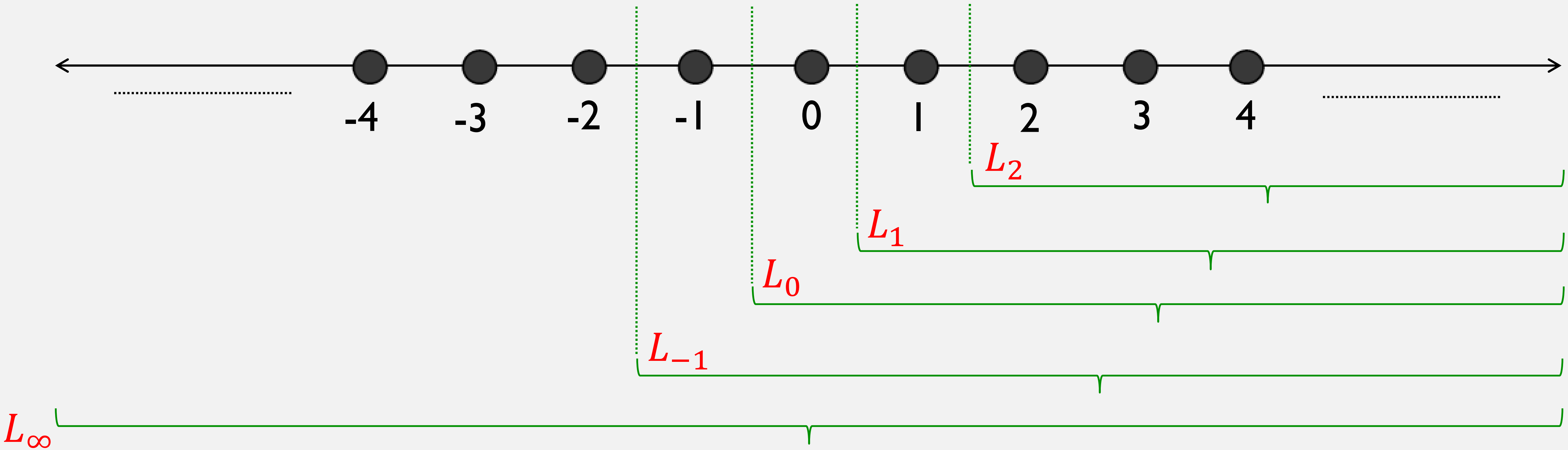
- Example: $\mathcal{C} = \{L_{\text{even integers}}, L_{\text{all integers}}\}$
- At each step, algorithm generates a new even integer…
- Generates correctly from $t = 1$, no matter the target language…

However, like identification, is generation in the limit also possible only for such simple collections?

HARD INSTANCE FOR IDENTIFICATION

$$L_\infty = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$
$$L_{\geq i} = \{i, i+1, i+2, i+3, \ldots\}$$

At each step, generate a number larger than any number seen as yet…



LANGUAGE GENERATION IN THE LIMIT

- <u>Theorem (Kleinberg-Mullainathan ’24):</u> *Every* countable collection of languages is generatable in the limit! 😲

- Includes finite, regular, context-free/sensitive, recursively enumerable, …. 🤯

- Recall that identifiability failed even for extremely simple collections… 🤦

LANGUAGE GENERATION IN THE LIMIT

- Known collection $\mathcal{C} = \{L_1, L_2, L_3, \dots\}$
- Adversary chooses some target language L_z, starts enumerating it

$$x_1, x_2, x_3, x_4, x_5, \dots\dots$$

- At each time step t, algorithm generates a string z_t
- Generates in the limit if beyond some large enough $t^\star$, all strings generated are new and in L_z

$t^\star$ can depend on target L_z as well as enumeration order! 😓

chooses L_{70}

😈	x_1	x_2	x_3	x_4	………	$x_{t^\star}$	$x_{t^\star+1}$ ………
😇	z_1	z_2	z_3	z_4	………	$z_{t^\star}$ new, $\in L_{70}$	$z_{t^\star+1}$ ……… new, $\in L_{70}$

LANGUAGE GENERATION IN THE LIMIT

- Limitation: Definition allows that the time step $t^\star$ beyond which algorithm generates validly can depend on the enumeration order!
- Example: Suppose $\mathcal{C} = \{L_1, L_2\}$

$$L_1 = \{\dots, -3, -2, -1, 1, 2, 3, \dots\}$$

$$L_2 = \{0, 1, 2, 3, 4, \dots\}$$

Kleinberg-Mullainathan's algorithm also faces this issue 😓

- Suppose L_2 is the target language, but adversary enumerates it as

$$1, 2, 3, 4, 5, 6, \dots$$

- Natural algorithm: generate from first consistent language in collection
- Until adversary shows 0, can keep generating negative numbers from L_1 😓

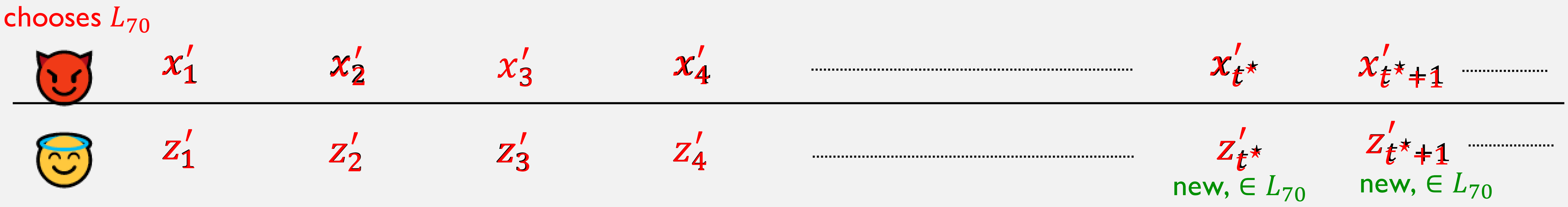
NON-UNIFORM GENERATION IN THE LIMIT

(Li, Raman, Tewari '24)

- $\mathcal{C} = \{L_1, L_2, L_3, \dots\}$
- Adversary chooses some target language L_z, starts enumerating it in an order of their choosing

$$x_1, x_2, x_3, x_4, x_5, \dots\dots$$

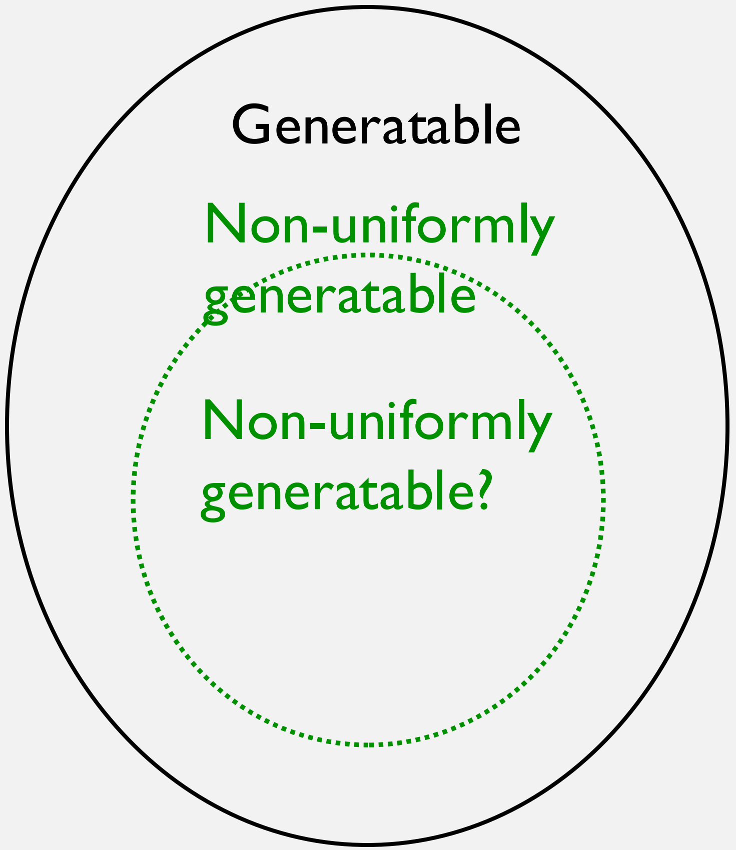
- At each time step t, algorithm generates a string z_t
- Non-uniformly generates in the limit if the moment the algorithm sees $t^\star = t^\star(\mathcal{C}, L_z)$ distinct strings , all strings generated thereafter are new and in L_z



NON-UNIFORM GENERATION IN THE LIMIT

- Open Question (Li, Raman, Tewari '24): Is every countable collection of languages non-uniformly generatable in the limit?

- Theorem (Charikar, P '24): Yes! 🙃



Countable language collections

also concurrently resolved by Li, Raman, Tewari

NON-UNIFORM GENERATION ALGORITHM

<u>Algorithm:</u>

1) At time step t, consider the languages

$$L_1, L_2, \dots, L_t$$

2) Say L_i is the first language consistent with the input seen so far; initialize $I_t = L_i$

3) For any subsequent language L_j that is also consistent with the input:

If $|I_t \cap L_j| = \infty$, update $I_t = I_t \cap L_j$

Else move on leaving I_t unaffected

4) Generate arbitrary new string from I_t

<u>Example:</u>

Suppose $S_5 = \{x_1, x_2, x_3, x_4, x_5\}$

L_1	L_2	L_3	L_4	L_5
$S_5 \subseteq L_1$	$S_5 \not\subseteq L_2$	$S_5 \subseteq L_3$	$S_5 \subseteq L_4$	$S_5 \not\subseteq L_5$
$I_5 = L_1$	$I_5 = L_1$	$\lvert I_5 \cap L_3 \rvert < \infty$ $I_5 = L_1$	$\lvert I_5 \cap L_4 \rvert = \infty$ $I_5 = L_1 \cap L_4$	$I_5 = L_1 \cap L_4$

Therefore, at $t = 5$, algorithm generates a string from $I_5 = L_1 \cap L_4$

NON-UNIFORM GENERATION ALGORITHM

Algorithm:

1) At time step t, consider the languages

$$L_1, L_2, \ldots, L_t$$

2) Say L_i is the first language consistent with the input seen so far; initialize $I_t = L_i$

3) For any subsequent language L_j that is also consistent with the input:

If $|I_t \cap L_j| = \infty$, update $I_t = I_t \cap L_j$

Else move on leaving I_t unaffected

4) Generate arbitrary new string from I_t

Invariant: I_t from which string is generated is always infinite

Suppose target language is L_z

$$\{L_1, L_2, \ldots L_z, \ldots\}$$

Observation:

1) L_z is always consistent with input

2) Beyond $t = z$, L_z is always under consideration

Only want that when L_z is encountered, $I_t \cap L_z = \infty$

NON-UNIFORM GENERATION ALGORITHM

Key Definition (Non-uniform Complexity):

For any language $L_i \in \mathcal{C}$, define its non-uniform complexity $m(L_i)$ as follows:

$m(L_i) =$ maximum over subsets of $\{L_1, \dots, L_i\}$ that contain L_i and have finite intersection

Example:

$L_1 \quad L_2 \quad L_3 \quad L_4 \quad L_5$

Suppose $|L_3 \cap L_1| = \infty, |L_3 \cap L_2| = 100, |L_3 \cap L_2 \cap L_1| = 95$

Then $m(L_3) = \max\{|L_3 \cap L_2|, |L_3 \cap L_2 \cap L_1|\} = \max\{100, 95\} = 100$

$m(L_i)$ = maximum over subsets of $\{L_1, \ldots, L_i\}$ that contain L_i and have finite intersection

<u>Algorithm:</u>

1) At time step t, consider the languages

$$L_1, L_2, \ldots, L_t$$

2) Say L_i is the first language consistent with the input seen so far; initialize $I_t = L_i$

3) For any subsequent language L_j that is also consistent with the input:

If $|I_t \cap L_j| = \infty$, update $I_t = I_t \cap L_j$

Else move on leaving I_t unaffected

4) Generate arbitrary new string from I_t

<u>Claim:</u> Consider $\mathcal{C} = \{L_1, L_2, \ldots L_z, \ldots\}$

$$t^\star(L_z, \mathcal{C}) = \max(z, m(L_z) + 1)$$

non-uniform guarantee! 🥳

<u>Proof:</u>

Consider t satisfying $|S_t| \geq t^\star(L_z, \mathcal{C})$

L_z under consideration since $t \geq z$

Suppose L_z did not get added to I_t

$L_1 \quad \ldots\ldots \quad L_z \quad \ldots\ldots \quad L_t$

$$|S_t| \leq I_t \cap L_z = |\underset{\supseteq S_t}{L_1} \cap \underset{\supseteq S_t}{L_{10}} \cap \cdots \cap \underset{\supseteq S_t}{L_{z-1}} \cap \underset{\supseteq S_t}{L_z}| < \infty$$

But $|S_t| \geq m(L_z) + 1 \Rightarrow\Leftarrow$

NON-UNIFORM GENERATION WITH MEMBERSHIP QUERIES

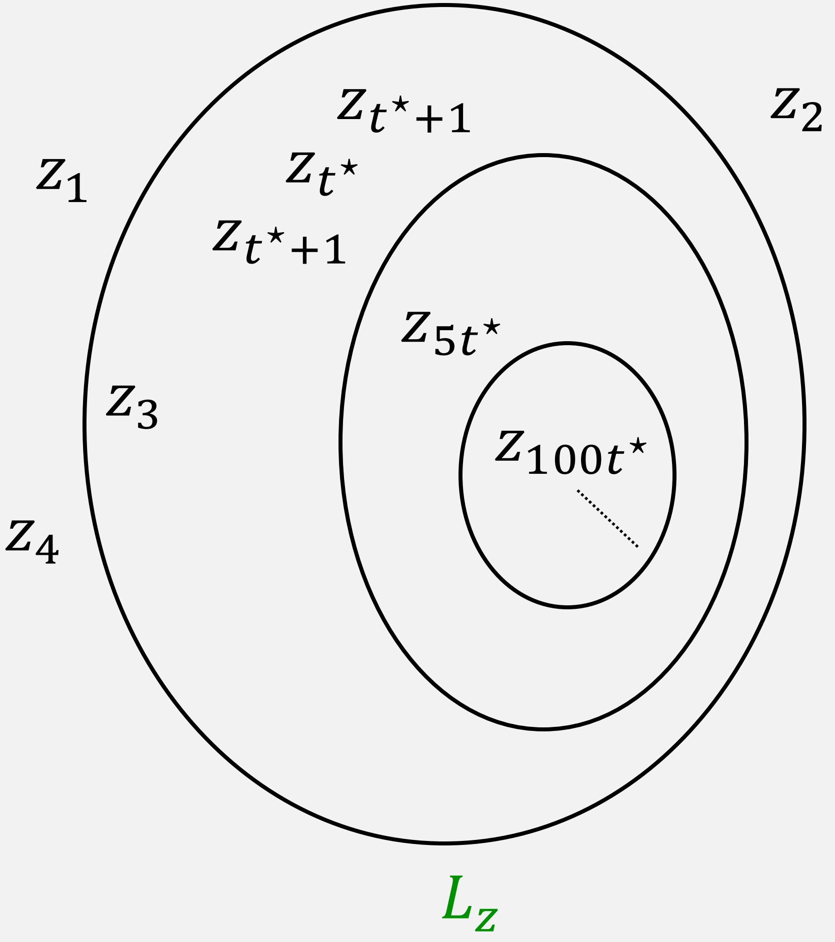
- Our non-uniform generation algorithm requires access to an oracle that, given any finite subcollection of languages, responds with whether the intersection of languages in the subcollection is finite or not
- Kleinberg-Mullainathan's algorithm requires only a membership query oracle, that answers queries of the form "is z in L_i?"
- Can we get non-uniform generation for all countable collections with only membership queries?
- <u>Theorem (Charikar, P '24):</u> Any algorithm that non-uniformly generates from all collections of size 2 cannot be solely implemented with membership queries

"non-uniform generation provably requires stronger oracles"

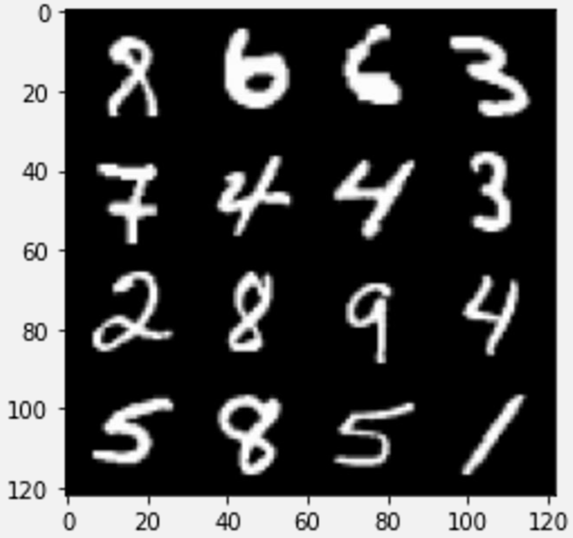
KLEINBERG-MULLAINATHAN'S ALGORITHM

Property: Lack of breadth

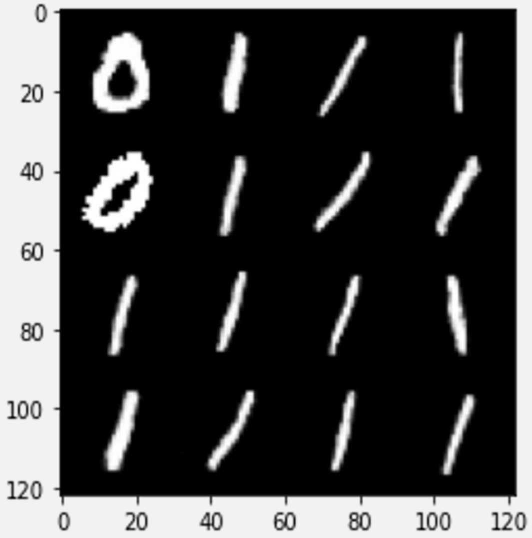
1) Algorithm starts off by producing invalid strings for a while
2) Eventually, it refines its hallucinations, and produces only valid strings thereafter
3) As t increases, algorithm potentially generates from an increasingly small subset of L_z



VALIDITY - BREADTH TRADEOFF



Train Data Point



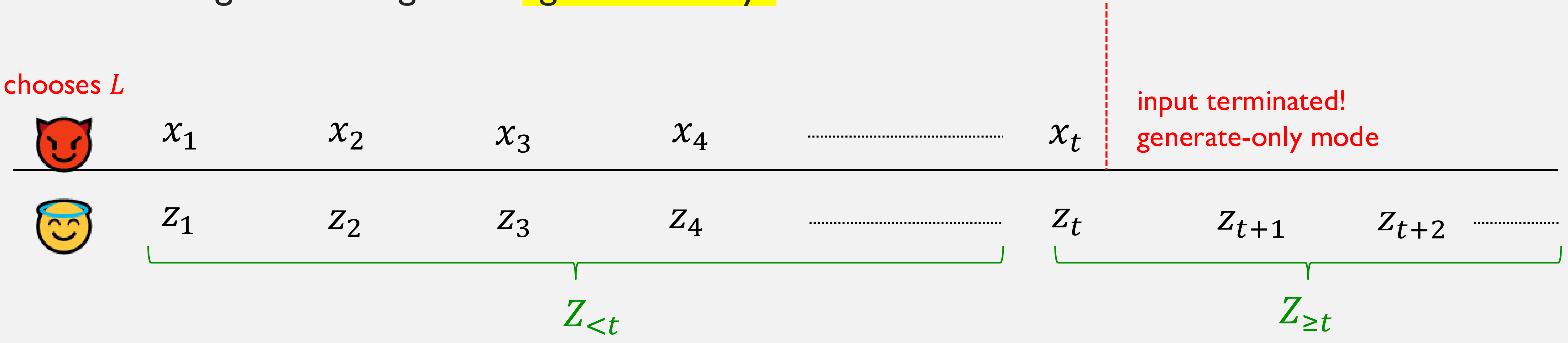
GAN Generated Data Point

Mode Collapse in GANs

Is the validity-breadth tradeoff fundamental to language generation in the limit?
Or can we come up with other algorithms that get the best of both worlds?

EXHAUSTIVE GENERATION

- Recall that the input eventually contains every string from the target language
- What if we can terminate the input at any time, and ask the generating algorithm to go into "generate-only" mode?



1) Want $Z_{<t} \cup Z_{\geq t}$ to cover the target language 2) Want $Z_{\geq t}$ to be valid strings

EXHAUSTIVE GENERATION

chooses L

$$\overbrace{x_1 \quad x_2 \quad x_3 \quad x_4 \quad \cdots \quad x_t}^{S_t}$$

input terminated!
generate-only mode

$$\underbrace{z_1 \quad z_2 \quad z_3 \quad z_4 \quad \cdots}_{Z_{<t}} \quad \underbrace{z_t \quad z_{t+1} \quad z_{t+2} \quad \cdots}_{Z_{\geq t}}$$

- Exhaustively generates in the limit if for all t beyond some large enough $t^\star$

1) (Validity) $|Z_{\geq t} \setminus L| < \infty$ "stops hallucinating eventually"

2) (Breadth) $S_t \cup Z_{<t} \cup Z_{\geq t} \supseteq L$ "covers all of K eventually"

EXHAUSTIVE GENERATION

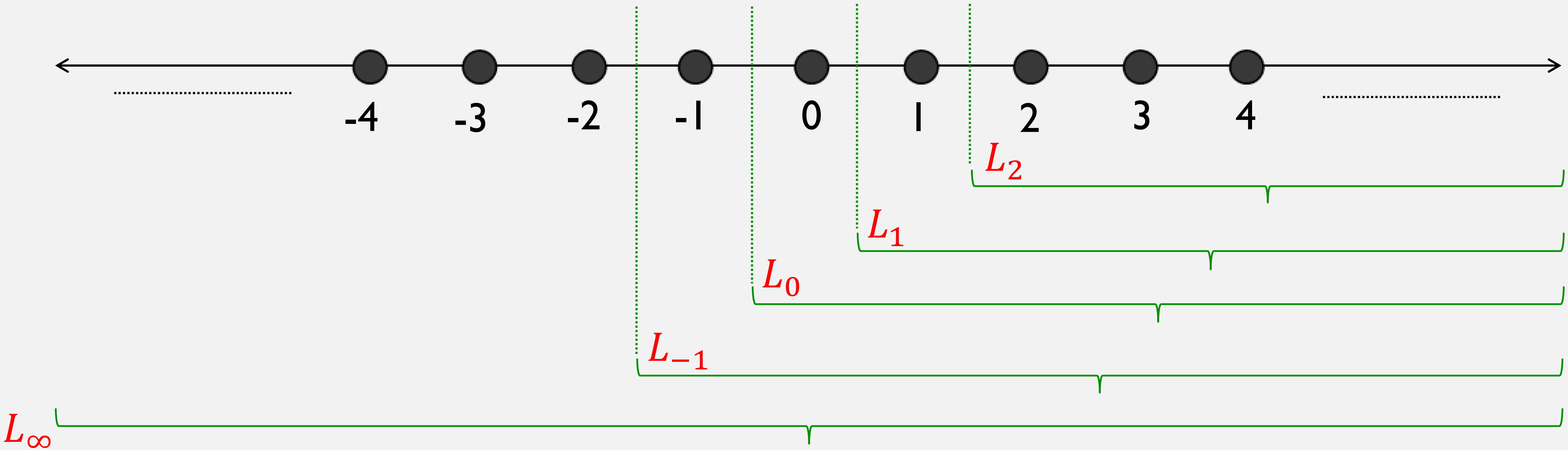
- Recall: *Every* countable collection can be generated in the limit…

- Theorem (Charikar, P '24): There exists a simple countable collection that cannot be exhaustively generated in the limit

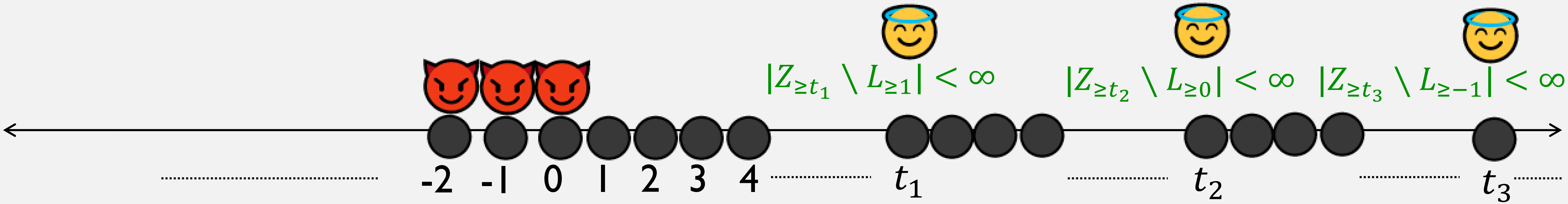
- Indicates that validity-breadth tradeoff is real in a formal sense for language generation in the limit
- Adds to growing evidence in literature that language models with desirable properties must hallucinate (Kalai-Vempala '24, Xu-Jain-Kankanalli '24, etc.)

HARD INSTANCE FOR IDENTIFICATION

$$L_\infty = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$
$$L_{\geq i} = \{i, i+1, i+2, i+3, \dots\}$$



EXHAUSTIVE GENERATION LOWER BOUND



Valid enumeration of $L_{\geq 1}$; at some finite time t_1, algorithm must exhaustively generate $L_{\geq 1}$

Valid enumeration of $L_{\geq 0}$; at some finite time t_2, algorithm must exhaustively generate $L_{\geq 0}$

Valid enumeration of $L_{\geq -1}$; at some finite time t_3, algorithm must exhaustively generate $L_{\geq -1}$

Infinite sequence $t_1 < t_2 < t_3 < \cdots$ such that at t_i, $\left|Z_{\geq t_i} \setminus L_{\geq 2-i}\right| < \infty$

However, adversary has produced a valid enumeration of L_∞

There must exist t_∞ such that for $t \geq t_\infty$, $Z_{<t} \cup S_t \cup Z_{\geq t} \supseteq L_\infty$

EXHAUSTIVE GENERATION CHARACTERIZATION

- However, identifiability $\ncong$ exhaustive generation!
- <u>Example:</u> $L_\infty =$ all integers

  $L_{-i} =$ all integers except i
- <u>Algorithm:</u> simply start generating $0, -1, 1, -2, 2, -3, 3, -4, 4, \ldots\ldots$
- Recall: A collection is identifiable iff it satisfies Angluin's condition..
- <u>Theorem (Charikar, P '24):</u> A collection is exhaustively generatable iff it satisfies a weaker version of Angluin's condition
- Precise characterization of exhaustive generation! 😇

parallel work by Kalavasis-Mehrotra-Velegkas '24 also proposes a definition of breadth; stronger than ours; closer to identifiability

GENERATION WITH FEEDBACK

- What if at each step t, algorithm can ask "does y_t belong to the target language?"

chooses L

	x_1	x_2	x_3	x_4		$x_{t^\star}$	$x_{t^\star+1}$
	$y_1$in L?	$y_2$in L?	$y_3$in L?	$y_4$in L?		$y_{t^\star}$in L?	$y_{t^\star+1}$in L?
	No!	No!	Yes!	No!		Yes!	No!
	z_1	z_2	z_3	z_4		$z_{t^\star}$ new, $\in L$	$z_{t^\star+1}$ new, $\in L$

- We characterize this setting with an abstract complexity parameter of the collection!

SUMMARY

- All countable collections can be non-uniformly generated!
- Lower bound for non-uniform generation with only membership queries!
- Exhaustive Generation: Validity-Breadth tradeoff necessary in generation
- Characterization of Exhaustive Generation
- Characterization of Generation with Feedback

Going forward . . . .

- What if input strings have noise? Noise models for generation [Raman, Raman '25]
- Notions of qualitative diversity in generated strings [Peale, Raman, Reingold '25]