On the Limits of Language Generation:
Trade-Offs Between Hallucination and Mode Collapse

Alkis Kalavasis

Anay Mehrotra

Grigoris Velegkas

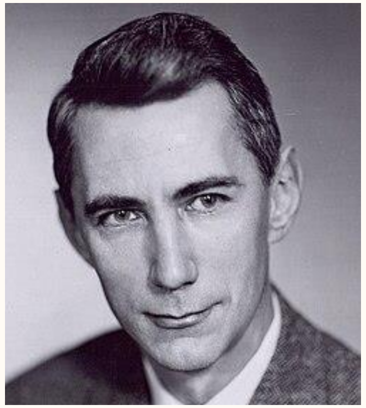
57th Annual ACM Symposium on Theory of Computing – 26 June 2025

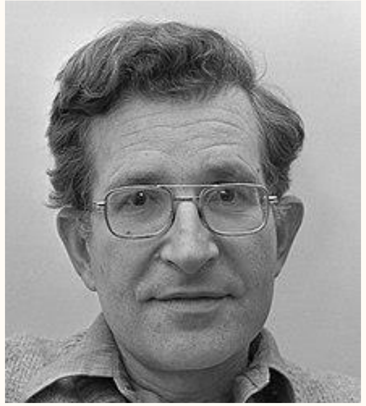
Outline of the Talk

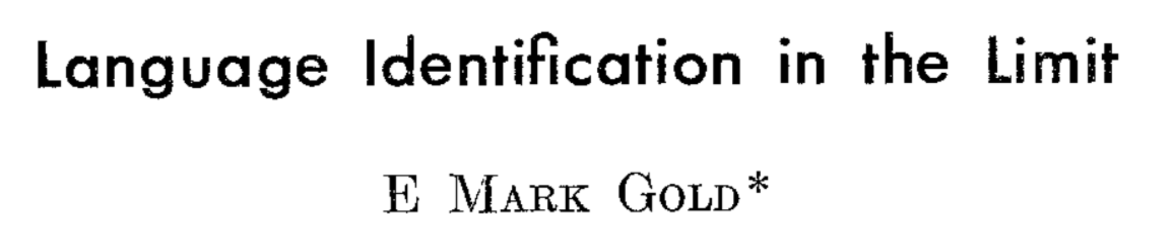
1. Introduction
   a. CS and Language Generation
   b. Language Generation in the Limit
2. Our Model
3. Our Results
4. Technical Overview
5. Future Work

CS and Language Generation

Computer scientists have been fascinated by language acquisition by humans and machines for decades





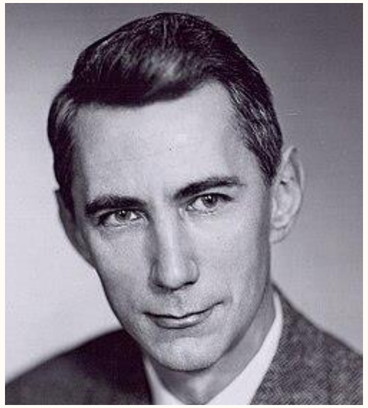


Language Identification in the Limit

E MARK GOLD*

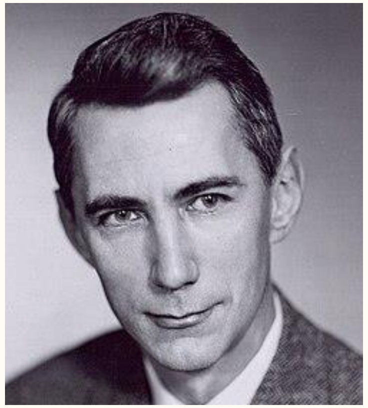
CS and Language Generation

1951 Shannon
Prediction and entropy of English



CS and Language Generation

1951 Shannon
Prediction and entropy of English

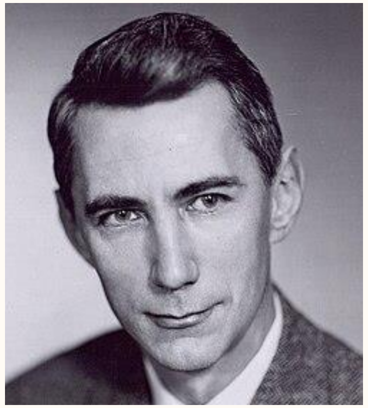


- A generation game between Betty and Claude Shannon

```
(1) THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG
(2) ----ROO------NOT-V-----I------SM----OBL----
(1) READING LAMP ON THE DESK SHED GLOW ON
(2) REA----------O------D----SHED-GLO--O--
(1) POLISHED WOOD BUT LESS ON THE SHABBY RED CARPET
(2) P-L-S-----O---BU--L-S--O------SH-----RE--C------
```

CS and Language Generation

1951 Shannon
Prediction and entropy of English



- Introduced *n*-grams – had tremendous impact in the 1980s!

2-gram model
Rhodesian Army offensive on average salary increase it four networks …

5-gram model
He praised love's ability to use dialogue to effect an emotional response…

CS and Language Generation

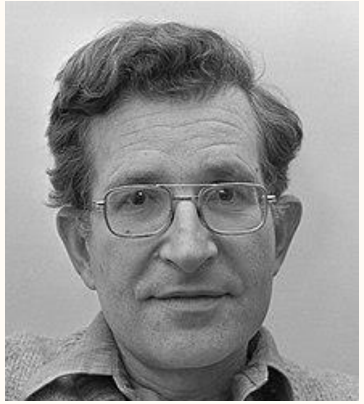
1951 Shannon
Prediction and entropy of English

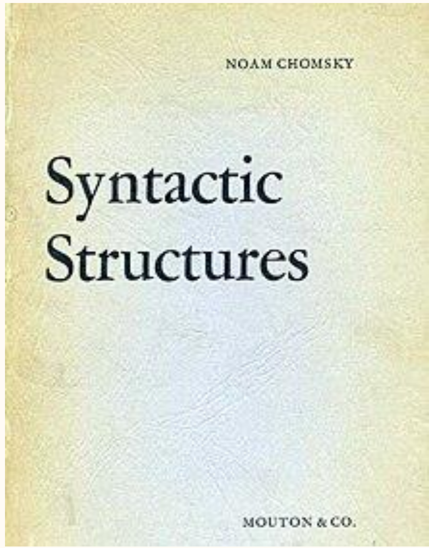
CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars



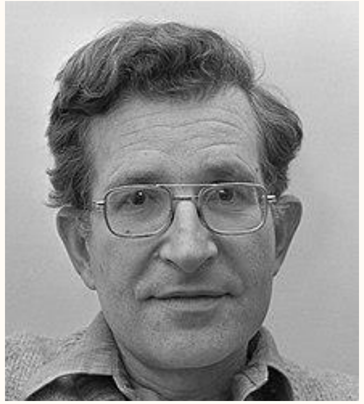


CS and Language Generation

1951 Shannon
Prediction and entropy of English

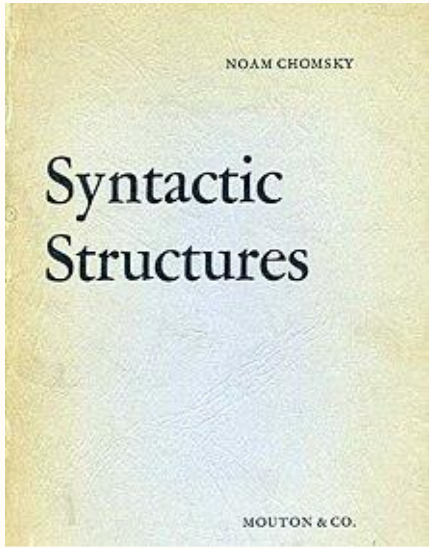
→

1957 Chomsky
Syntactic structures & formal grammars



- Separated grammar (syntax) and semantics

Colorless green ideas sleep furiously

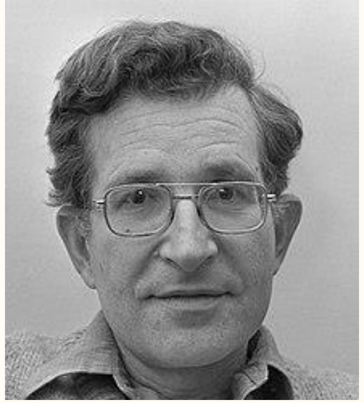


CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

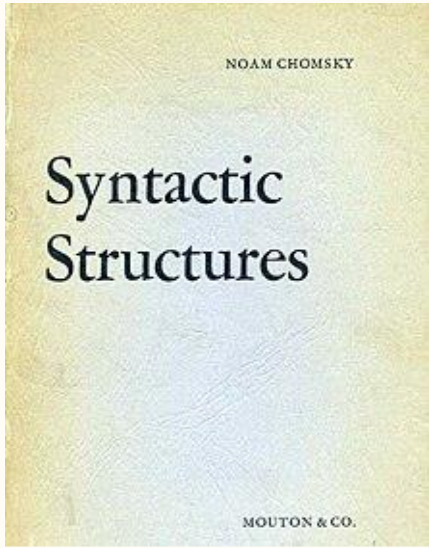
1957 Chomsky
Syntactic structures & formal grammars



- Separated grammar (syntax) and semantics

Colorless green ideas sleep furiously

- Introduced a hierarchy of grammars
- Apart from linguistics also influenced TOC



CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

I wish to construct a precise model for "able to speak English"…
to investigate theoretically how it can be achieved artificially

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

I wish to construct a precise model for "able to speak English"…
to investigate theoretically how it can be achieved artificially

Since we cannot explicitly write down the rules of English…
artificial intelligence… will have to learn… from implicit data…

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

I wish to construct a precise model for "able to speak English"…
to investigate theoretically how it can be achieved artificially

Since we cannot explicitly write down the rules of English…
artificial intelligence… will have to learn… from implicit data…

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

I wish to construct a precise model for "able to speak English"…
to investigate theoretically how it can be achieved artificially

Since we cannot explicitly write down the rules of English…
artificial intelligence… will have to learn… from implicit data…

learning from samples!

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

CS and Language Generation

1951 Shannon *Prediction and entropy of English*	→	1957 Chomsky *Syntactic structures & formal grammars*	→	1967 Gold *Formal language identification model*

- **Laid the groundwork** for the celebrated PAC framework [Valiant, 1984] (Turing Award, 2010)

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

- **Laid the groundwork** for the celebrated PAC framework [Valiant, 1984] (Turing Award, 2010)

- Contains many ideas developed much later in learning theory
  → Learning from samples,
  → Hypothesis class,
  → Two-player online games, and even active learning!

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

- Also had a significant impact in *linguists*
  - → Do inductive biases of humans help them learn to speak?
  - → Do children need interaction to learn to speak?
  - → …

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

CS and Language Generation

1951 Shannon
Prediction and entropy of English

1957 Chomsky
Syntactic structures & formal grammars

1967 Gold
Formal language identification model

1970s - 1980s
Early conversation systems - SHRDLU! & IBM Watson

CS and Language Generation

- **1951 Shannon** *Prediction and entropy of English*
- **1957 Chomsky** *Syntactic structures & formal grammars*
- **1967 Gold** *Formal language identification model*
- **1970s - 1980s** *Early conversation systems - SHRDLU! & IBM Watson*
- **1990s - 2000s** *Wide-spread use of statistical models: RNNs…*

CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars

→

1967 Gold
Formal language identification model

→

1970s - 1980s
Early conversation systems - SHRDLU! & IBM Watson

→

1990s - 2000s
Wide-spread use of statistical models: RNNs…

→

2010s – Present
- *word-to-vec'13*
- *Attention'14*
- *seq-seq tran..'14*
- *Transformers'17*
- *GPT2 Paper'19..*

Modern Language Generators – LLMs

Modern Language Generators – LLMs

I am giving a talk about language generation. Can you write something brief (it has to go on a slide) and creative to demonstrate what today's language models are capable of?

Reasoned about language model capabilities for 5 seconds >

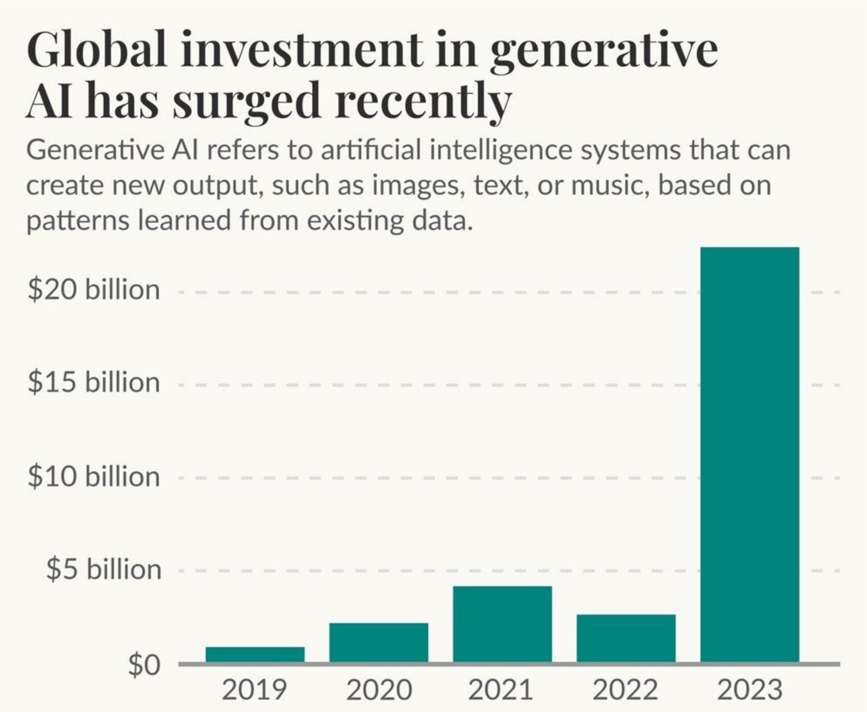
Language Generation Magic

From code to poetry, AI weaves words into art. For example:

> "In circuits' hum, a dream takes flight—
> A digital muse sparks worlds of light.
> Infinite stories in a single line,
> Where thought and code beautifully entwine."

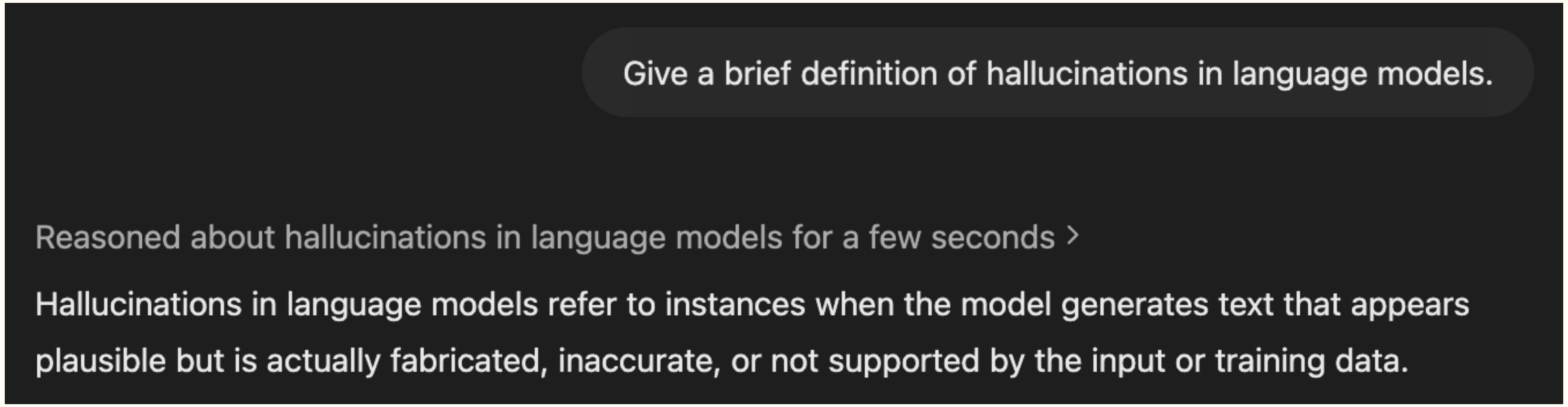
This is today's language model: creativity, computation, and endless possibilities.

Modern Language Generators – LLMs



Source: Quid via AI Index (2024) and US Bureau of Labor Statistics (2024)

Hallucinations in LLMs



Hallucinations in LLMs

Hallucinations in language models refer to instances when the model generates text that appears plausible but is actually fabricated, inaccurate, or not supported by the input or training data.

Early days … "simple" errors

How many Rs in the word strawberry?

There are **two** Rs in the word "strawberry."

Hallucinations in LLMs

Hallucinations in language models refer to instances when the model generates text that appears plausible but is actually fabricated, inaccurate, or not supported by the input or training data.

Subsequently … hallucinations in products

Google still recommends glue for your pizza

/ It's almost like AI answers aren't fully baked!

by Elizabeth Lopatto

Jun 11, 2024, 6:24 PM EDT

Hallucinations in LLMs

Hallucinations in language models refer to instances when the model generates text that appears plausible but is actually fabricated, inaccurate, or not supported by the input or training data.

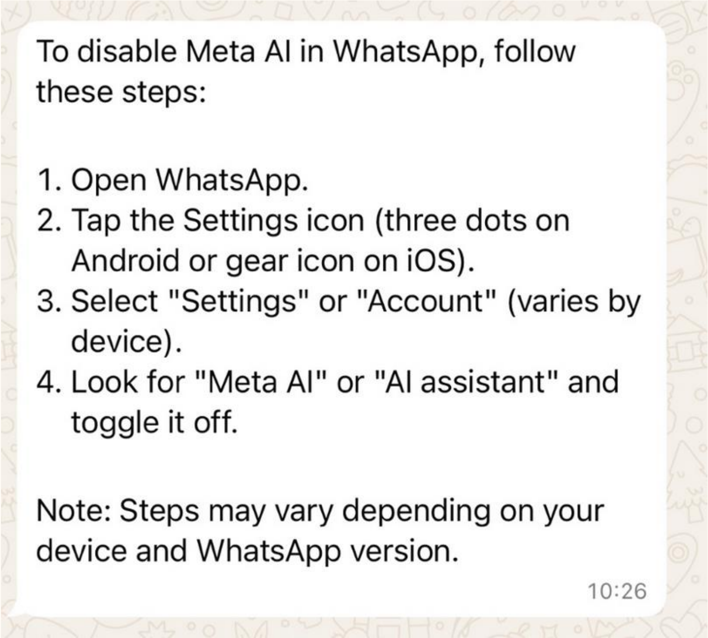
Subsequently … hallucinations in products

Source: Twitter / X

Hallucinations in LLMs

Hallucinations in language models refer to instances when the model generates text that appears plausible but is actually fabricated, inaccurate, or not supported by the input or training data.

Subsequently … hallucinations in products



Source: Twitter / X

Hallucinations in LLMs

Hallucinations in language models refer to instances when the model generates text that appears plausible but is actually fabricated, inaccurate, or not supported by the input or training data.

Today, hallucinations rare due to innovations (*e.g., chain of thought*)
Yet models *still hallucinate* on complex tasks

Hallucinations in LLMs

Hallucinations in language models refer to instances when the model generates text that appears plausible but is actually fabricated, inaccurate, or not supported by the input or training data.

Today, hallucinations rare due to innovations (*e.g., chain of thought*)
Yet models *still hallucinate* on complex tasks

Easy to avoid hallucinations by *limiting* the range or breadth of the model

Question. Can *hallucinations* be avoided while retaining *breadth* via better (but "*similar*") models and training methods?

Outline of the Talk

1. Introduction
   a. CS and Language Generation
   b. Language Generation in the Limit
2. Our Model
3. Our Results
4. Technical Overview
5. Future Work

Language Generation in the Limit

A model by Kleinberg and Mullainathan (NeurIPS, 2024)

Language Generation in the Limit

Jon Kleinberg
Departments of Computer Science
and Information Sciene
Cornell University
Ithaca NY

Sendhil Mullainathan
Booth School of Business
University of Chicago
Chicago IL

Language Identification in the Limit [Gold, 1967]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \dots\}$

Language Identification in the Limit [Gold, 1967]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \dots\}$

E.g., regular languages

Language Identification in the Limit [Gold, 1967]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Game between adversary and learner

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$

Language Identification in the Limit [Gold, 1967]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Game between adversary and learner

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t

Language Identification in the Limit [Gold, 1967]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Game between adversary and learner

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \ldots$

Language Identification in the Limit [Gold, 1967]

Game between adversary and learner

1. Adversary picks target $K = L_{i^\star}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, i^\star, i^\star, i^\star, \ldots$

Language Identification in the Limit [Gold, 1967]

Game between adversary and learner

1. Adversary picks target $K = L_{i^\star}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, i^\star, i^\star, i^\star, \ldots$

Adversary has to present a complete enumeration

Example: $K = \mathbb{N}$,

Language Identification in the Limit [Gold, 1967]

Game between adversary and learner

1. Adversary picks target $K = L_{i^\star}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, i^\star, i^\star, i^\star, \ldots$

Adversary has to present a complete enumeration

Example: $K = \mathbb{N}, \quad 2, 4, 6, \ldots,$

Language Identification in the Limit [Gold, 1967]

Game between adversary and learner

1. Adversary picks target $K = L_{i^\star}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, i^\star, i^\star, i^\star, \ldots$

Adversary has to present a complete enumeration

Example: $K = \mathbb{N}$, $\quad 2, 4, 6, \ldots, \quad 1, 2, 3, \ldots$

Language Identification in the Limit [Gold, 1967]

Game between adversary and learner

1. Adversary picks target $K = L_{i^\star}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, i^\star, i^\star, i^\star, \ldots$

Adversary has to present a complete enumeration

Example: $K = \mathbb{N}$, $2, 4, 6, \ldots,$ ✗ $1, 2, 3, \ldots$ ✓ and $2, 4, 6, \ldots, 1, 2, 3, \ldots$ ✗

Language Identification in the Limit [Gold, 1967]

Game between adversary and learner

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \ldots$

Learners access:

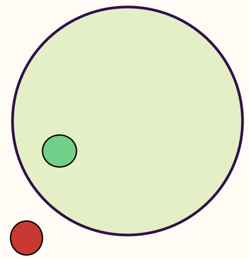
Language Identification in the Limit [Gold, 1967]

Game between adversary and learner

1. Adversary picks target $K = L_{i^\star}$
2. Rounds $t = 1, 2, 3, \dots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\dots, i_t, i^\star, i^\star, i^\star, \dots$

Learners access:

Is $w \in L_i$?
Membership Query



Language Identification in the Limit [Gold, 1967]

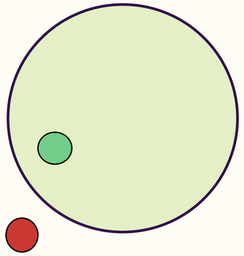
Game between adversary and learner

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) learner guesses target-index i_t
3. Learner wins if guesses are eventually right: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{\ldots}$

Learners access:

Is $w \in L_i$?

Membership Query



Language Generation in the Limit [KM'24]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Language Generation in the Limit [KM'24]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \dots\}$

Game between adversary and generator

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$

Language Generation in the Limit [KM'24]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Game between adversary and generator

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) generator outputs *unseen* string g_t

Language Generation in the Limit [KM'24]

- Domain $\mathcal{X}$, e.g., $\{a\text{-}z, A\text{-}Z\}^*$ or $\mathbb{N}$
- Collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

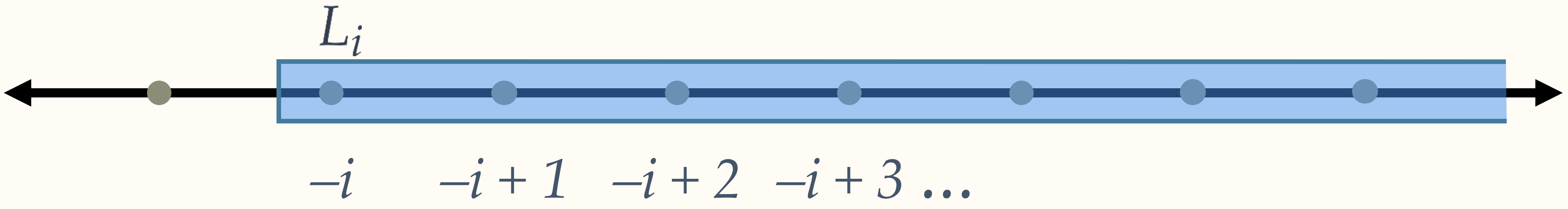
Game between adversary and generator

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) adversary shows example $x_t \in K$, and
   (b) generator outputs *unseen* string g_t
3. Generator wins if guesses are eventually in K: $\boldsymbol{K \ni g_t, g_{t+1}, \ldots}$

   after some finite time $t < \infty$

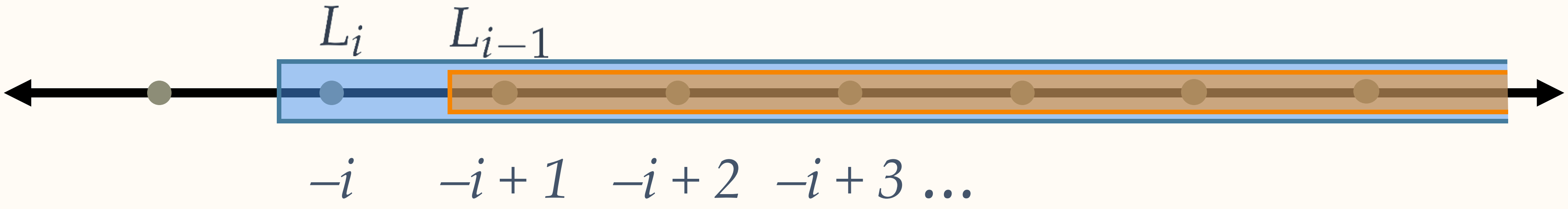
Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\ .$$



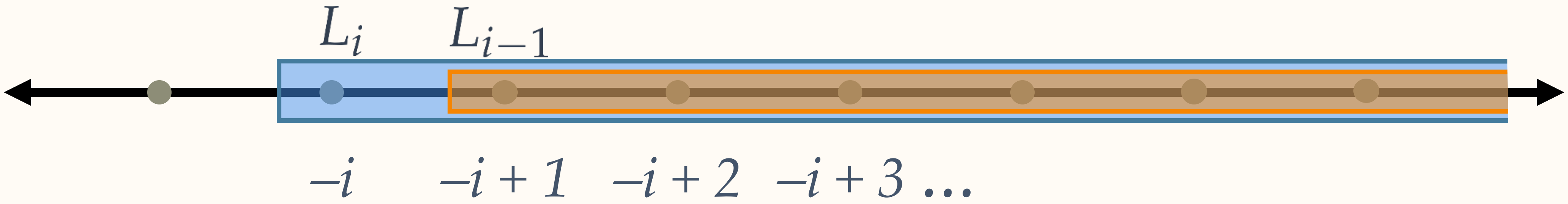
Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\,.$$



Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

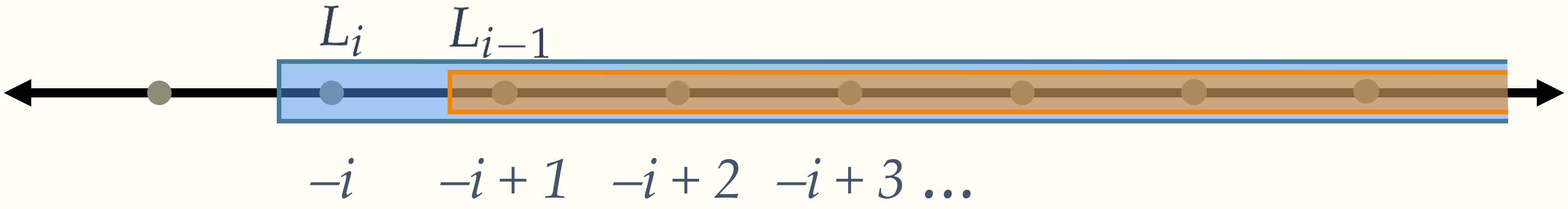
$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\,.$$



▷ Is $\mathcal{L}$ generatable?

Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

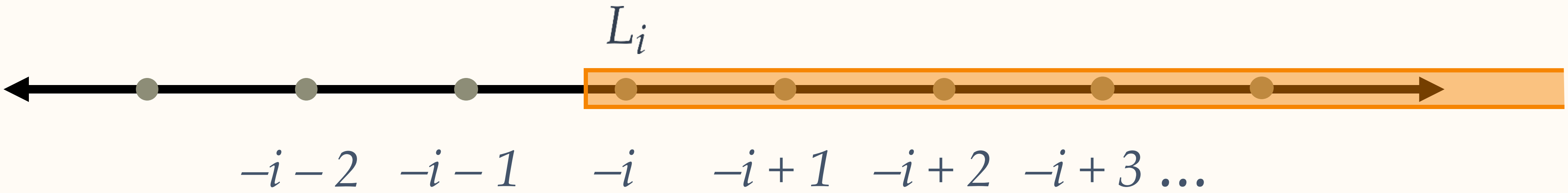
$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\,.$$



▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_1+2, \ldots\}$

Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \dots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \dots\}\,.$$

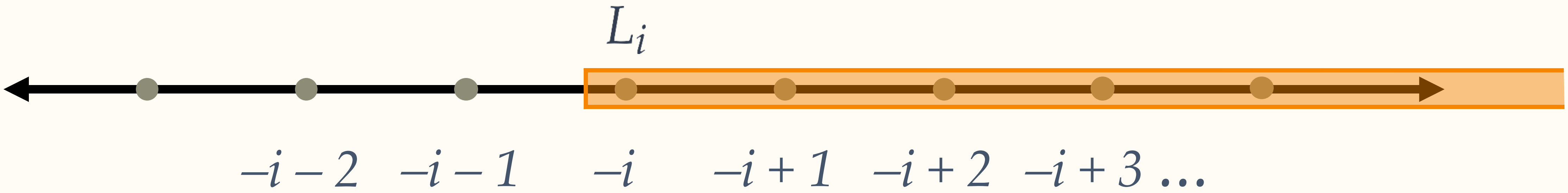


▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!

Output an unseen example from $\{x_1+1, x_2+2, \dots\}$

Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \dots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \dots\}.$$

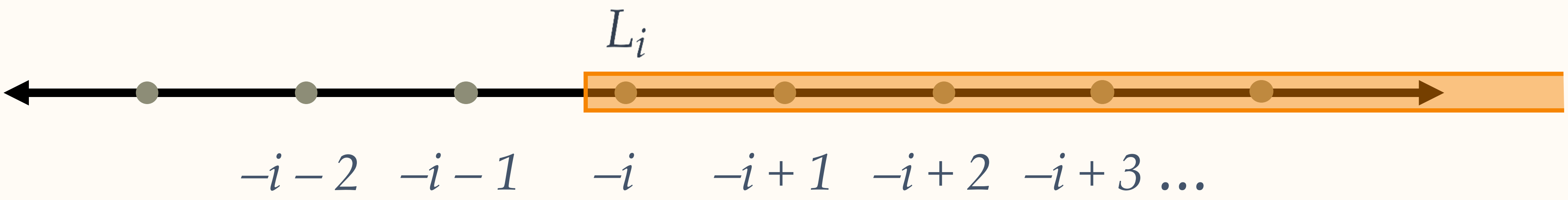


▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \dots\}$

▷ Is $\mathcal{L}$ identifiable?

Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\,.$$



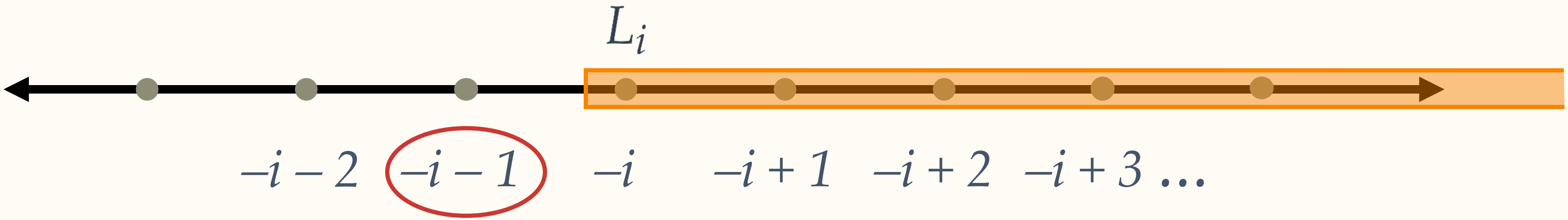
▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \ldots\}$

▷ Is $\mathcal{L}$ identifiable?

Theorem. Angluin (1980) $\mathcal{L}$ is not identifiable

Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \dots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \dots\}\,.$$



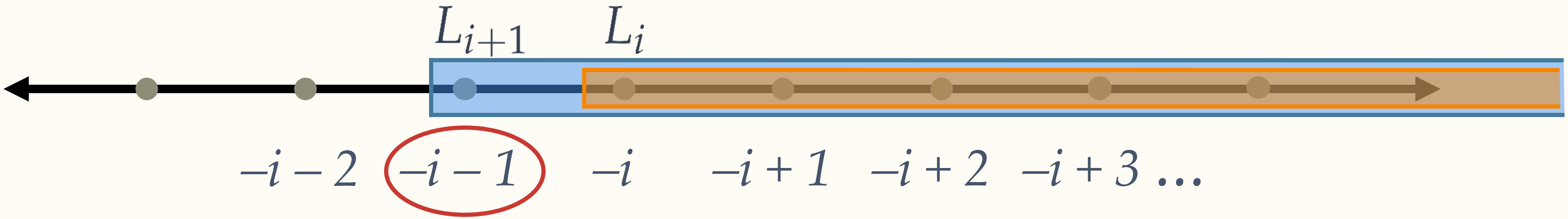
▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \dots\}$

▷ Is $\mathcal{L}$ identifiable?

Theorem. Angluin (1980) $\mathcal{L}$ is not identifiable

Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\,.$$



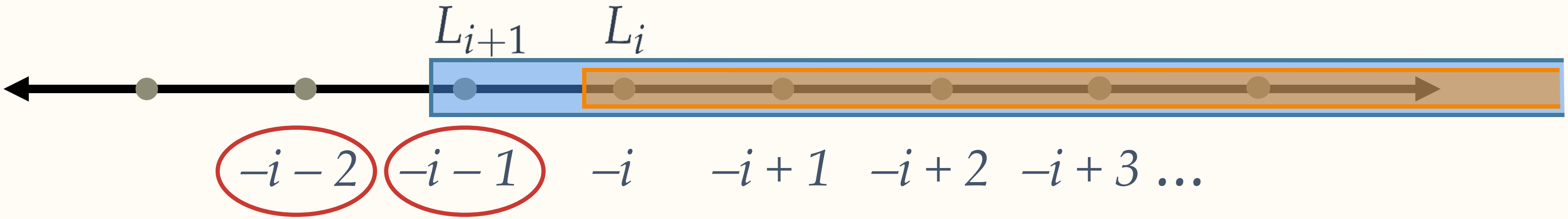
▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \ldots\}$

▷ Is $\mathcal{L}$ identifiable?

Theorem. Angluin (1980) $\mathcal{L}$ is not identifiable

Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\ .$$



▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \ldots\}$

▷ Is $\mathcal{L}$ identifiable?

Theorem. Angluin (1980) $\mathcal{L}$ is not identifiable

Key Properties of the Model

- It is a prompt-less model

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM’24]

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM’24]
- Abstracts away – training process, next-token-prediction, …

For positive results these impose important limitations …[Bhattamishra, Ahuja, and Goyal’20] [Sanford, Hsu, Telgarsky’23] [Peng, Narayanan, and Papadimitriou’24] [Chen, Peng, and Wu’24]…

We focus on negative results – which show that the source of “difficulty” are not these details

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM'24]
- Abstracts away – training process, next-token-prediction, …
- Practical data isn't *adversarial* – it is drawn from a *distribution*

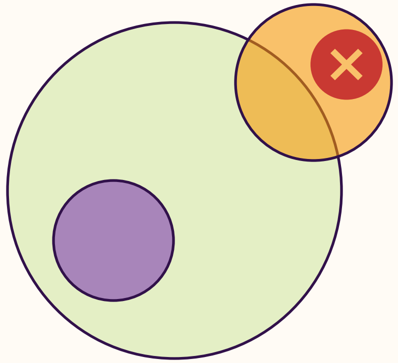
We'll consider a *distributional model*

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM'24]
- Abstracts away – training process, next-token-prediction, …
- Practical data isn't *adversarial* – we'll study a *distributional model*
- Do not need to learn *all* the language – does not capture *breadth*

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM'24]
- Abstracts away – training process, next-token-prediction, …
- Practical data isn't *adversarial* – we'll study a *distributional model*
- Do not need to learn *all* the language – does not capture *breadth*



Target language K Training data S Generator's support G

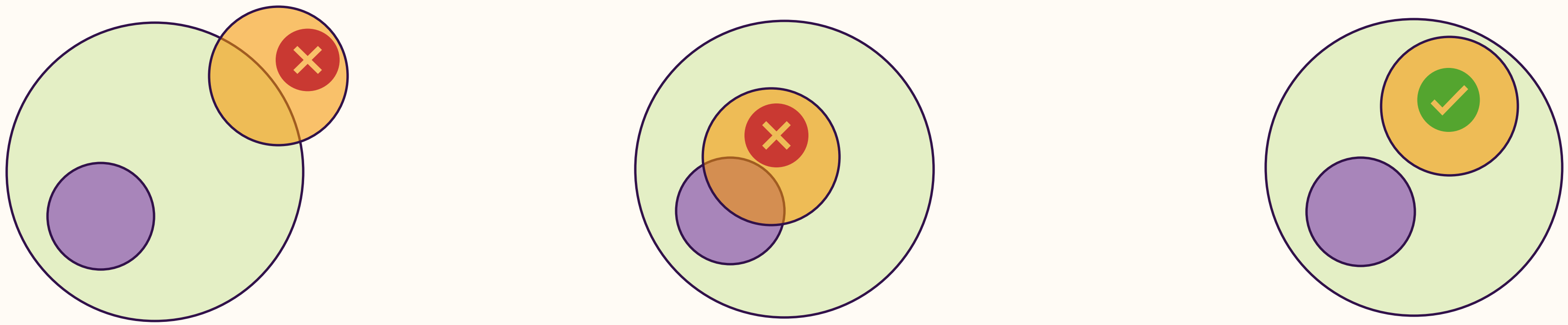
Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM'24]
- Abstracts away – training process, next-token-prediction, …
- Practical data isn't *adversarial* – we'll study a *distributional model*
- Do not need to learn *all* the language – does not capture *breadth*

Target language K Training data S Generator's support G

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM'24]
- Abstracts away – training process, next-token-prediction, …
- Practical data isn't *adversarial* – we'll study a *distributional model*
- Do not need to learn *all* the language – does not capture *breadth*



Target language K Training data S Generator's support G

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM'24]
- Abstracts away – training process, next-token-prediction, …
- Practical data isn't *adversarial* – we'll study a *distributional model*
- Does not capture *breadth* – we'll study *generation with breadth*

Key Properties of the Model

- ~~It is a prompt-less model~~ – can be extended [KM'24]
- Abstracts away – training process, next-token-prediction, …
- Practical data isn't *adversarial* – we'll study a *distributional model*
- Does not capture *breadth* – we'll study *generation with breadth*

Question (also asked by [KM'24]). Is it possible to achieve consistent language generation with breadth or is there some inherent trade-off between consistency and breadth?

Outline of the Talk

1. Introduction
2. Our Model
    a. Statistical Model of Language Generation
    b. Notions of Breadth
    c. MOP Assumption
3. Our Results
4. Technical Overview
5. Future Work

Statistical Model

- Statistical model of identification and generation based on [Angluin, 1988], [BHMvHY, 2021]

Statistical Model

- Statistical model of identification and generation based on [Angluin, 1988], [BHMvHY, 2021]

- Adversary chooses K and *distribution over* K

Statistical Model

- Statistical model of identification and generation based on [Angluin, 1988], [BHMvHY, 2021]

- Adversary chooses K and *distribution over* K

- For $t = 1,2,\ldots$ generator draws t i.i.d. samples and outputs $\mathcal{G}_t \subseteq \mathcal{X}$

Statistical Model

- Statistical model of identification and generation based on [Angluin, 1988], [BHMvHY, 2021]
- Adversary chooses K and *distribution over* K
- For $t = 1,2,\ldots$ generator draws t i.i.d. samples and outputs $\mathcal{G}_t \subseteq \mathcal{X}$
- Generator needs to satisfy some (binary) *property* P
  - Unified treatment of identification and generation

Statistical Model

- Statistical model of identification and generation based on [Angluin, 1988], [BHMvHY, 2021]

- Adversary chooses K and *distribution over K*

- For $t = 1,2,\ldots$ generator draws t i.i.d. samples and outputs $\mathcal{G}_t \subseteq \mathcal{X}$

- Generator needs to satisfy some (binary) *property P*
  - Unified treatment of identification and generation

- As t increases does $\mathbb{E}[P(\mathcal{G}_t)] \to 0$ and how quickly?

Statistical Model

- Domain $\mathcal{X}$, collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Statistical Model

- Domain $\mathcal{X}$, collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Game between adversary and generator

1. Adversary picks target $\boldsymbol{K} = \boldsymbol{L_{i^\star}}$ and distribution over K
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) generator draws t i.i.d. examples from the distribution and outputs $\mathcal{G}_t \subseteq \mathcal{X}$.
3. Generator wins if it satisfies the property P as $t \to \infty$: $\lim_{t\to\infty} \mathbb{E}[P(\mathcal{G}_t)] = 0$

Statistical Model

- Domain $\mathcal{X}$, collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$

Game between adversary and generator

1. Adversary picks target $\boldsymbol{K = L_{i^\star}}$ and distribution over K
2. Rounds $t = 1, 2, 3, \ldots,$
   (a) generator draws t i.i.d. examples from the distribution and outputs $\mathcal{G}_t \subseteq \mathcal{X}$.
3. Generator wins if it satisfies the property P as $t \to \infty$: $\lim_{t\to\infty} \mathbb{E}[P(\mathcal{G}_t)] = 0$
4. Generator has rate R if $\mathbb{E}[P(\mathcal{G}_t)] \leq C \cdot R(c \cdot t)$ for all t, for distribution-dependent c, C

Rate of Convergence

- An important rate function is the exponential $R(t) = e^{-t}$

Rate of Convergence

- An important rate function is the exponential $R(t) = e^{-t}$

- Generator achieves P at exponential rate if for *every* K in the collection, and for *every distribution* over K, there exist C, c:

$$\mathbb{E}[P(\mathcal{G}_t)] \leq C \cdot e^{-c \cdot t}$$

Rate of Convergence

- An important rate function is the exponential $R(t) = e^{-t}$
- Generator achieves P at exponential rate if for *every* K in the collection, and for *every distribution* over K, there exist C, c:

$$\mathbb{E}[P(\mathcal{G}_t)] \leq C \cdot e^{-c \cdot t}$$

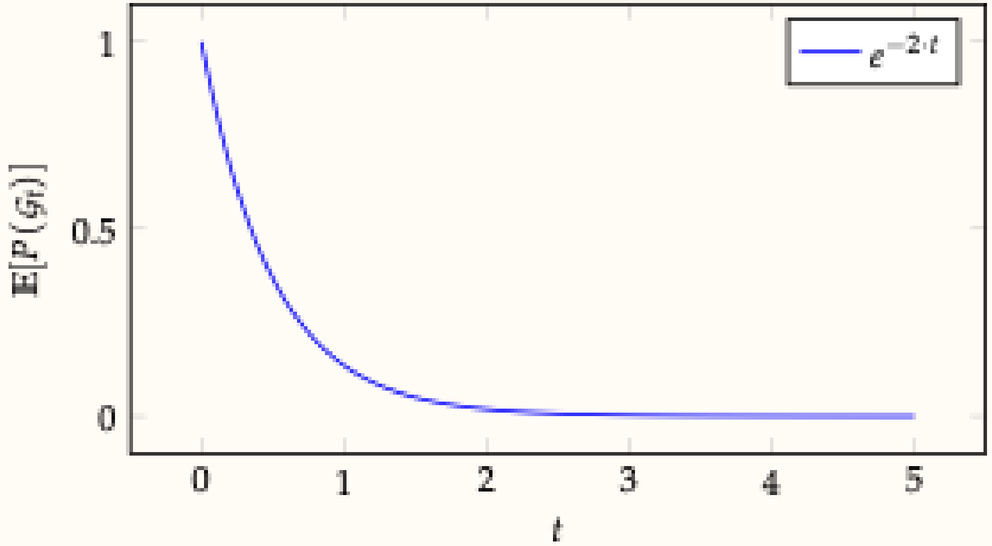
- Order of quantifiers is crucial!

Rate of Convergence

- An important rate function is the exponential $R(t) = e^{-t}$
- Generator achieves P at exponential rate if for *every* K in the collection, and for *every distribution* over K, there exist C, c:

$$\mathbb{E}[P(\mathcal{G}_t)] \leq C \cdot e^{-c \cdot t}$$

- Order of quantifiers is crucial!



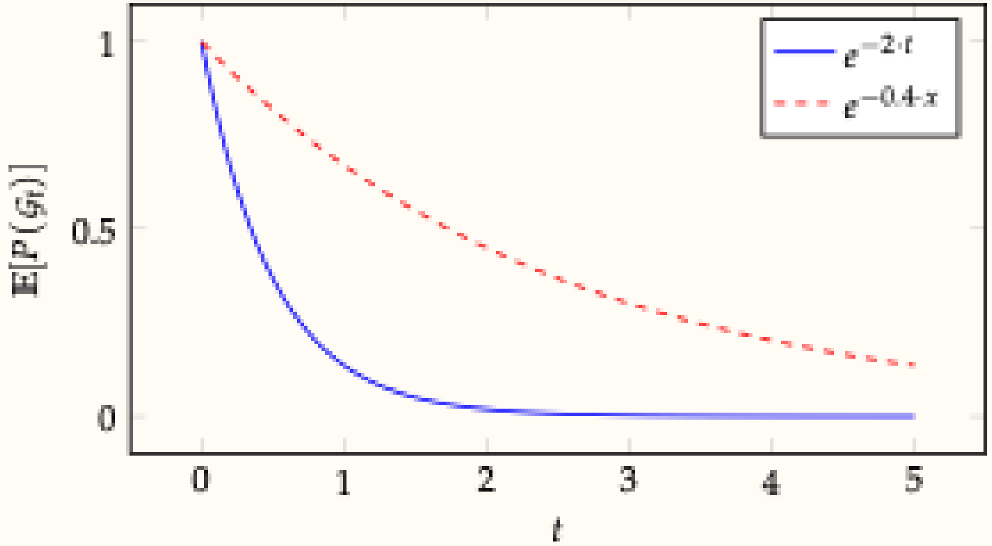
Rate of Convergence

- An important rate function is the exponential $R(t) = e^{-t}$

- Generator achieves P at exponential rate if for *every* K in the collection, and for *every distribution* over K, there exist C, c:

$$\mathbb{E}[P(\mathcal{G}_t)] \leq C \cdot e^{-c \cdot t}$$

- Order of quantifiers is crucial!



Rate of Convergence

- An important rate function is the exponential $R(t) = e^{-t}$
- Generator achieves P at exponential rate if for *every* K in the collection, and for *every distribution* over K, there exist C, c:

$$\mathbb{E}[P(\mathcal{G}_t)] \leq C \cdot e^{-c \cdot t}$$

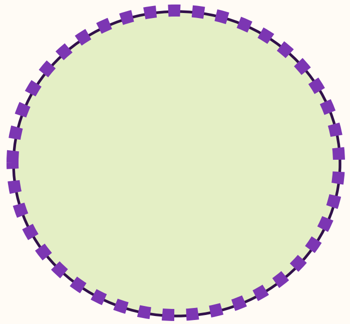
- Order of quantifiers is crucial!
- If $\lim_{t\to\infty} \mathbb{E}[P(\mathcal{G}_t)] \neq 0$ for some distribution no rate is achievable

Outline of the Talk

1. Introduction
2. Our Model
    a. Statistical Model of Language Generation
    b. Notions of Breadth
    c. MOP Assumption
3. Our Results
4. Technical Overview
5. Future Work

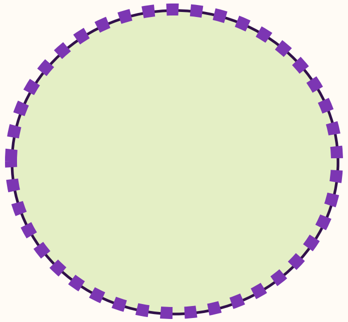
Notions of Breadth

(Exact) breadth:
Contain *all* elements of K, *nothing* outside of it

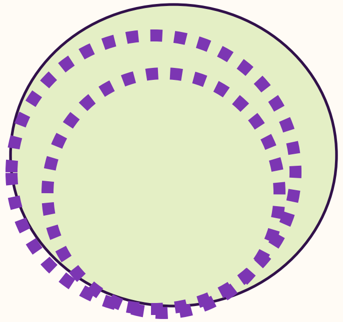


Notions of Breadth

(Exact) breadth:
Contain *all* elements of K, *nothing* outside of it

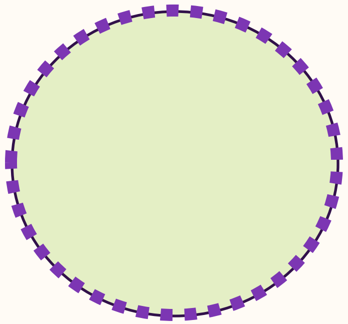


Approximate breadth:
Miss finitely many elements of K, has *nothing* outside of it

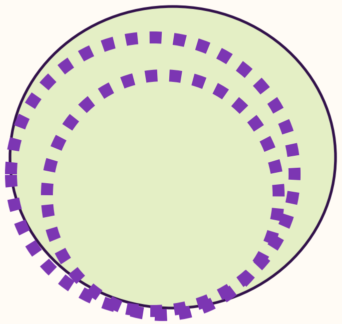


Notions of Breadth

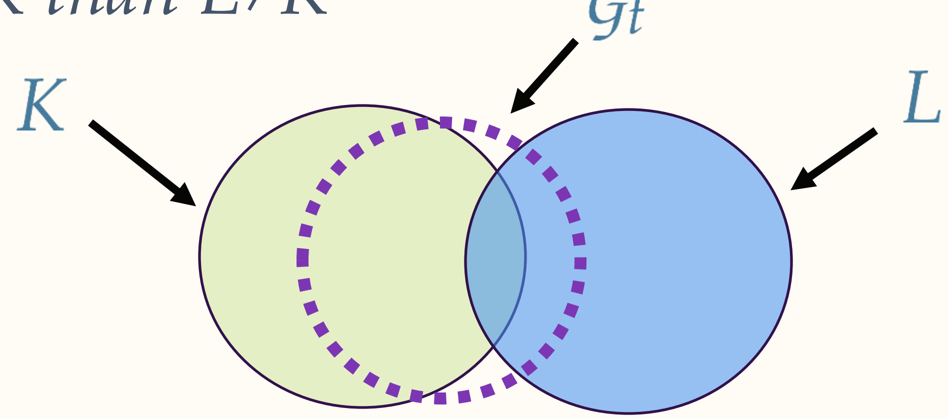
(Exact) breadth:
Contain *all* elements of K, *nothing* outside of it



Approximate breadth:
Miss finitely many elements of K, has *nothing* outside of it



Unambiguous generation:
Output *closer (wrt symmetric difference)* to K *than* $L \neq K$



Outline of the Talk

1. Introduction
2. Our Model
    a. Statistical Model of Language Generation
    b. Notions of Breadth
    c. MOP Assumption
3. Our Results
4. Technical Overview
5. Future Work

MOP Assumption

- For our lower bounds we assume the generator satisfies the Membership Oracle Property (MOP):

MOP Assumption

- For our lower bounds we assume the generator satisfies the Membership Oracle Property (MOP):
  - For any string x we can decide if x is in the output of the generator

MOP Assumption

- For our lower bounds we assume the generator satisfies the Membership Oracle Property (MOP):

    - For any string x we can decide if x is in the output of the generator

    - Mild assumption, satisfied by large class of generators including *next-token-predictors*

MOP Assumption

- For our lower bounds we assume the generator satisfies the Membership Oracle Property (MOP):
    - For any string x we can decide if x is in the output of the generator
    - Mild assumption, satisfied by large class of generators including *next-token-predictors*
    - For certain generators it might be undecidable (related to the halting problem)

Outline of the Talk

1. Introduction
2. Our Model
3. Our Results
    a. Main Result
    b. Outline of Proof and Challenges
    c. Further Results
4. Technical Overview
5. Future Work

Our Results

Language generation with breadth (statistical) is equivalent to
language identification in the limit (adversarial)
for any "usual" generators

Main Result

Main Theorem [This work]. For any language collection $\mathcal{L}$:

- If $\mathcal{L}$ is not identifiable, no generator $\mathcal{G}$ with decidable MOP can generate from $\mathcal{L}$ with breadth (at any rate).
- If $\mathcal{L}$ is identifiable, there is $\mathcal{G}$ with decidable MOP, which generates with breadth from $\mathcal{L}$ at (near) exponential rate.

Main Result

Main Theorem [This work]. For any language collection $\mathcal{L}$:

- ▷ If $\mathcal{L}$ is not identifiable, no generator $\mathcal{G}$ with decidable MOP can generate from $\mathcal{L}$ with breadth (at any rate).
- ▷ If $\mathcal{L}$ is identifiable, there is $\mathcal{G}$ with decidable MOP, which generates with breadth from $\mathcal{L}$ at (near) exponential rate.

Contrast with generation without breadth:

Theorem [This work]. For any $\mathcal{L}$, there is a $\mathcal{G}$ (with decidable MOP) that generates from $\mathcal{L}$ (without breadth) at *exponential rate*.

Main Result

Main Theorem [This work]. For any language collection $\mathcal{L}$:

- ▷ If $\mathcal{L}$ is not identifiable, no generator $\mathcal{G}$ with decidable MOP can generate from $\mathcal{L}$ with breadth (at any rate).
- ▷ If $\mathcal{L}$ is identifiable, there is $\mathcal{G}$ with decidable MOP, which generates with breadth from $\mathcal{L}$ at (near) exponential rate.

Analogous characterizations for the other two notions of breadth – generators are required to have decidable MOP and be *"stable"*

Main Result

Main Theorem [This work]. For any language collection $\mathcal{L}$:

- ▷ If $\mathcal{L}$ is <u>not identifiable</u>, no generator $\mathcal{G}$ with decidable MOP can generate from $\mathcal{L}$ with breadth (at any rate).
- ▷ If $\mathcal{L}$ is <u>identifiable</u>, there is $\mathcal{G}$ with decidable MOP, which generates with breadth from $\mathcal{L}$ at (near) exponential rate.

- The above characterization also extends to *all* generators [Charikar and Pabbaraju, COLT'25] [Kalavasis, Mehrotra, Velegkas, arXiv'24]
- [Peale, Raman, Reingold, ICML'25] [Kleinberg and Wei, arXiv'25] study finer-grained characterizations (in online setting)

(Short) Overview of Proof of Main Result

Claim 1. If $\mathcal{L}$ is not identifiable, no generator $\mathcal{G}$ with decidable MOP can generate from $\mathcal{L}$ with breadth (at any rate).

Natural Strategy: Convert a generator $\mathcal{G}$ with breadth, to an identifier

Observation: Need to use some *property* of $\mathcal{G}$; otherwise, it only provides an enumeration of K that we already had!

Technical Vignette (Properties of $\mathcal{G}$)

1. $\mathcal{G}$ is non-adaptive

   Simple collections $\mathcal{L}$ remain unidentifiable for many enumerations

2. $\mathcal{G}$ samples from a fixed distribution

   $\mathcal{L}$ remain unidentifiable w.r.t. fixed distribution [Angluin'88]

(Short) Overview of Proof of Main Result

Idea 1: We will use membership oracle access to $\mathrm{supp}(\mathcal{G})$

Roughly, membership to $\mathrm{supp}(\mathcal{G})$, provides membership to K

This is sufficient to get an identifier for $\mathcal{L}$ in the limit from $\mathcal{G}$

Challenges in statistical setting. Our hope is:

Convert generator with breadth at rate $R(\cdot)$ to identifier at rate $R(\cdot)$

We need an *identifier in the limit* for contradiction.

$\mathcal{L}$ not identifiable in the limit, may be identifiable at rate $R'(\cdot)$

This is true for binary classification [BHMvHY21]

(Short) Overview of Proof of Main Result

Collection $\mathcal{L}$ → Not identifiable (in limit)

Strengthens the results of [Angluin'88]

↓

Not identifiable *at any rate*

We convert a generator with breath at rate R to an identifier at rate R

↓

Not generatable with breath *at any rate*

(Short) Overview of Proof of Main Result

Identifiable (in limit) ← Collection $\mathcal{L}$ → Not identifiable (in limit)

Standard online-to-batch strategy fails:
1. No feedback to fix size of batches
2. Majority vote: K can occur @ many indices

We use growing batch-sizes + postprocessing to identify the smallest index of K

↓

Identifiable at (near) exponential rate

↓

Generatable with breath at (near) exponential rate

Strengthens the results of [Angluin'88]

↓

Not identifiable *at any rate*

We convert a generator with breath at rate R to an identifier at rate R

↓

Not generatable with breath *at any rate*

Further Results: Rates for Identification

Theorem [This work]. For any "non-trivial" collection $\mathcal{L}$:

- ▷ If $\mathcal{L}$ is identifiable in the limit, there is $\mathcal{G}$, which identifies $\mathcal{L}$ at (near) exponential rate.
- ▷ If $\mathcal{L}$ is not identifiable in the limit, no generator $\mathcal{G}$ can identify $\mathcal{L}$ at any rate.

Further Results: We achieve *exact* exponential rates in various special: such as, when $|\mathcal{L}| < \infty$ or one has stronger access to $\mathcal{L}$.

Further Results: Negative Examples Help

Theorem [This work]. For any collection $\mathcal{L}$, given positive and negative examples, there exists a generator which generates from $\mathcal{L}$ with breadth at exponential rate.

Reminiscent of RLHF, which encodes *negative information*

Further Results: Negative Examples Help

Theorem [This work]. For any collection $\mathcal{L}$, given positive and negative examples, there exists a generator which generates from $\mathcal{L}$ with breadth at exponential rate.

Indeed, proxies for negative examples are found useful in practice

NEURAL TEXT DEGENERATION WITH UNLIKELIHOOD TRAINING

Sean Welleck[1,2*] Ilia Kulikov[1,2*] Stephen Roller[2] Emily Dinan[2]
Kyunghyun Cho[1,2,3] & Jason Weston[1,2]

NEGATIVE DATA AUGMENTATION

Abhishek Sinha[1*] Kumar Ayush[1*] Jiaming Song[1*] Burak Uzkent[1] Hongxia Jin[2]
Stefano Ermon[1]

Further Results: Negative Examples Help

Theorem [This work]. For any collection $\mathcal{L}$, given positive and negative examples, there exists a generator which generates from $\mathcal{L}$ with breadth at exponential rate.

Indeed, proxies for negative examples are found useful in practice

NEURAL TEXT DEGENERATION WITH UNLIKELIHOOD TRAINING

Sean Welleck[1,2*] Ilia Kulikov[1,2*] Stephen Roller[2] Emily Dinan[2]
Kyunghyun Cho[1,2,3] & Jason Weston[1,2]

NEGATIVE DATA AUGMENTATION

Abhishek Sinha[1*] Kumar Ayush[1*] Jiaming Song[1*] Burak Uzkent[1] Hongxia Jin[2]
Stefano Ermon[1]

Empirical Challenge:

Can one extract negative examples from real-world data?

Proof Overview of Main Result

Exponential Rates for Identification

- General online-to-batch conversion due to [BHMvHY, 2021]

Exponential Rates for Identification

- General online-to-batch conversion due to [BHMvHY, 2021]
  - Estimate time t^* so that with t^* i.i.d. draws the learner wins in the "online" game

Exponential Rates for Identification

- General online-to-batch conversion due to [BHMvHY, 2021]
  - Estimate time t^* so that with t^* i.i.d. draws the learner wins in the "online" game
  - Split the input into t/t^* non-overlapping batches

Exponential Rates for Identification

- General online-to-batch conversion due to [BHMvHY, 2021]
  - Estimate time t^* so that with t^* i.i.d. draws the learner wins in the “online” game
  - Split the input into t/t^* non-overlapping batches
  - Run the online game on each batch independently

Exponential Rates for Identification

- General online-to-batch conversion due to [BHMvHY, 2021]
  - Estimate time t^* so that with t^* i.i.d. draws the learner wins in the "online" game
  - Split the input into t/t^* non-overlapping batches
  - Run the online game on each batch independently
  - "Aggregate" outputs of the games (e.g., majority vote)

Exponential Rates for Identification

- General online-to-batch conversion due to [BHMvHY, 2021]
  - ~~Estimate time t^* so that with t^* i.i.d. draws the learner wins in the "online" game~~

    Crucially relies on having feedback!

  - ~~Split the input into t/t^* non-overlapping batches~~

Exponential Rates for Identification

- General online-to-batch conversation due to [BHMvHY, 2021]
  - ~~Estimate time t^* so that with t^* i.i.d. draws the learner wins in the "online" game~~

Crucially relies on having feedback!

  - ~~Split the input into t/t^* non-overlapping batches~~
  - ~~Run the online game on each batch independently~~

Multiple copies of correct language, cannot immediately aggregate!

  - ~~"Aggregate" outputs of the games (e.g., majority vote)~~

Exponential Rates for Identification

- Modified online-to-batch conversion
  - Choose time $t^* = \omega(1)$
  - Split the input into t/t^* non-overlapping batches
    - Gives "almost"-exponential rates
  - Run the identification game on each batch independently
  - "Post-process" the outputs s.t. all correct guesses are same index
    - Take the majority vote of the indices

Main Result for Identification

Theorem [This work]. For any "non-trivial" collection $\mathcal{L}$:

- ▷ If $\mathcal{L}$ is <u>identifiable</u> in the limit, there is $\mathcal{G}$, which identifies $\mathcal{L}$ at (near) exponential rate. ✓
- ▷ If $\mathcal{L}$ is <u>not identifiable</u> in the limit, no generator $\mathcal{G}$ can identify $\mathcal{L}$ at any rate. Main challenge, different from [BHMvHY'21]

Main Result for Identification

Theorem [This work]. For any "non-trivial" collection $\mathcal{L}$:

- ▷ If $\mathcal{L}$ is <u>identifiable</u> in the limit, there is $\mathcal{G}$, which identifies $\mathcal{L}$ at (near) exponential rate. ✓
- ▷ If $\mathcal{L}$ is <u>not identifiable</u> in the limit, no generator $\mathcal{G}$ can identify $\mathcal{L}$ at any rate. Main challenge, different from [BHMvHY'21]

- Can get exact exponential rates, but not in a black-box way

Exponential Rates for Generation

- Previous modified approach does not work

Exponential Rates for Generation

- Previous modified approach does not work
  - Main obstacle: cannot black-box "aggregate" generators
    - Formal results [Hanneke, Karbasi, Mehrotra, Velegkas '25]

Exponential Rates for Generation

- Previous modified approach does not work
  - Main obstacle: cannot black-box "aggregate" generators
    - Formal results [Hanneke, Karbasi, Mehrotra, Velegkas '25]
- Solution: avoid the aggregation altogether and show that the (online) algorithm of [KM'24] gives exponential rates!
  - First such result in the universal rates line of work

Exponential Rates for Generation

- [KM'24] algorithm: Given $S_t = \{s_1, \ldots, s_t\}$ create set of "critical" languages, output from highest-indexed one

Exponential Rates for Generation

- [KM'24] algorithm: Given $S_t = \{s_1, \ldots, s_t\}$ create set of "critical" languages, output from highest-indexed one
- Critical languages: $\mathcal{C}_t(S_t) \subseteq \mathcal{L}_t = \{L_1, \ldots, L_t\}$ such that
  - Every language $L_i \in \mathcal{C}_t(S_t)$ is consistent with S_t
  - For all $j < i$ if $L_j \in \mathcal{C}_t(S_t)$ then $L_i \subseteq L_j$

Exponential Rates for Generation

- [KM'24] algorithm: Given $S_t = \{s_1, \ldots, s_t\}$ create set of "critical" languages, output from highest-indexed one

- Critical languages: $\mathcal{C}_t(S_t) \subseteq \mathcal{L}_t = \{L_1, \ldots, L_t\}$ such that
  - Every language $L_i \in \mathcal{C}_t(S_t)$ is consistent with S_t
  - For all $j < i$ if $L_j \in \mathcal{C}_t(S_t)$ then $L_i \subseteq L_j$

- Key property: For every K there exists $S^K = \{s_{i_1}, \ldots, s_{i_\ell}\} \subseteq K$ s.t. if $S^K \subseteq S_t$ then the algorithm generates correctly

Exponential Rates for Generation

- [KM'24] algorithm: Given $S_t = \{s_1, \ldots, s_t\}$ create set of "critical" languages, output from highest-indexed one
- Critical languages: $\mathcal{C}_t(S_t) \subseteq \mathcal{L}_t = \{L_1, \ldots, L_t\}$ such that
  - Every language $L_i \in \mathcal{C}_t(S_t)$ is consistent with S_t
  - For all $j < i$ if $L_j \in \mathcal{C}_t(S_t)$ then $L_i \subseteq L_j$
- Key property: For every K there exists $S^K = \{s_{i_1}, \ldots, s_{i_\ell}\} \subseteq K$ s.t. if $S^K \subseteq S_t$ then the algorithm generates correctly
- [This work]: Every such algorithm has exponential rates!

Outline of the Talk

1. Introduction
2. Our Model
3. Our Results
   a. Main Result
   b. Outline of Proof and Challenges
   c. Further Results
4. Technical Overview
5. Future Work

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation: Partial results [KMV'24]
   (b) Fine-grained trade-offs between hallucinations and breadth: Partial results [KMV'24],[CP'24],[KW'25]

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation: Partial results [KMV'24]
   (b) Fine-grained trade-offs between hallucinations and breadth: Partial results [KMV'24],[CP'24],[KW'25]
2. Allow generators to output multiple responses (could bypass many impossiblity results)

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation: Partial results [KMV'24]
   (b) Fine-grained trade-offs between hallucinations and breadth: Partial results [KMV'24],[CP'24],[KW'25]
2. Allow generators to output multiple responses (could bypass many impossiblity results)
3. Developing computationally efficient algorithms in more structured settings

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation: Partial results [KMV'24]
   (b) Fine-grained trade-offs between hallucinations and breadth: Partial results [KMV'24],[CP'24],[KW'25]
2. Allow generators to output multiple responses (could bypass many impossiblity results)
3. Developing computationally efficient algorithms in more structured settings
4. What other type of feedback is useful? Partial results [CP'24]

Tutorial on Language Generation

At COLT 2025, this summer!

Visit: LanguageGeneration.github.io

Organizers:

Moses Charikar
Stanford

Anay Mehrotra
Yale University

Chirag Pabbaraju
Stanford

Charlotte Peale
Stanford

Grigoris Velegkas
Yale → Google Research

Thank you!