Inherent Trade-offs in Language Generation

Anay Mehrotra
Yale University

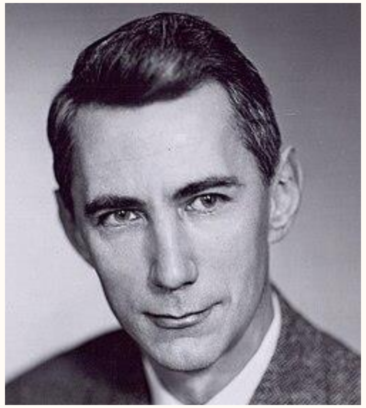
Alkis Kalavasis

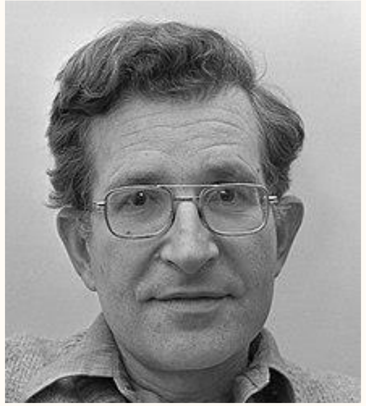
Grigoris Velegkas

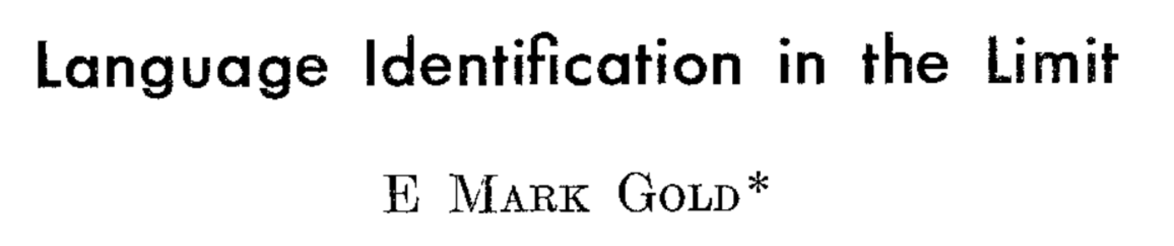
May 2025

CS and Language Generation

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



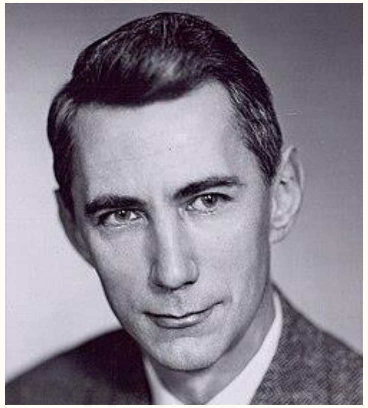



Language Identification in the Limit

E Mark Gold*

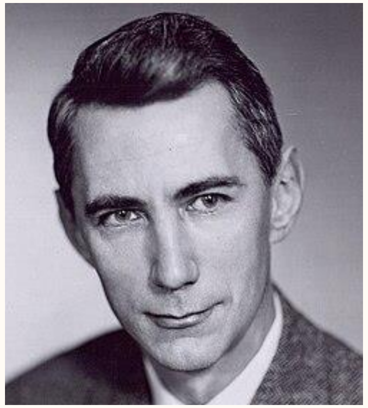
CS and Language Generation

1951 Shannon
Prediction and entropy of English



CS and Language Generation

1951 Shannon
Prediction and entropy of English

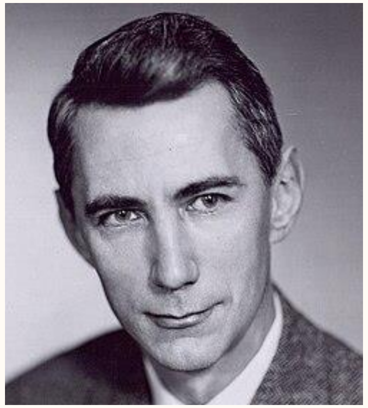


- A generation game between Betty and Claude Shannon

```
(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

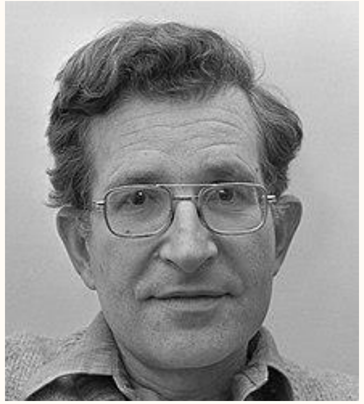
1951 Shannon
Prediction and entropy of English

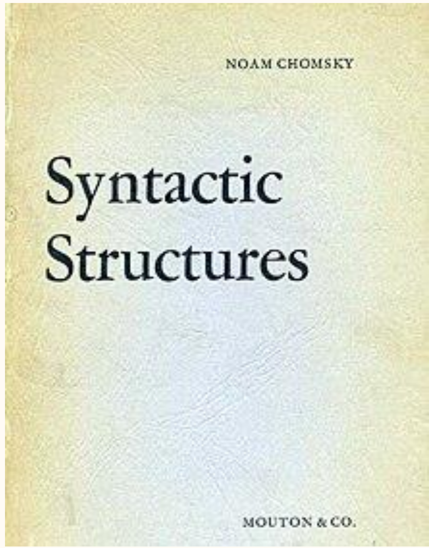
CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

1957 Chomsky
Syntactic structures & formal grammars



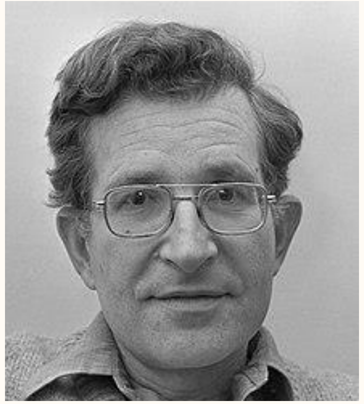


CS and Language Generation

1951 Shannon
Prediction and entropy of English

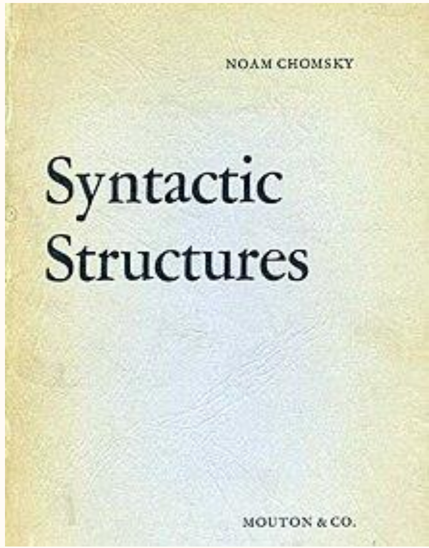
→

1957 Chomsky
Syntactic structures & formal grammars



- Separated grammar (syntax) and semantics

$\boxed{\text{Colorless green ideas sleep furiously}}$

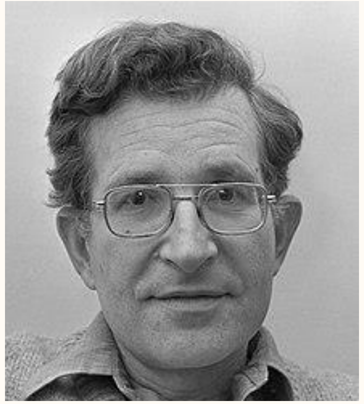


CS and Language Generation

1951 Shannon
Prediction and entropy of English

→

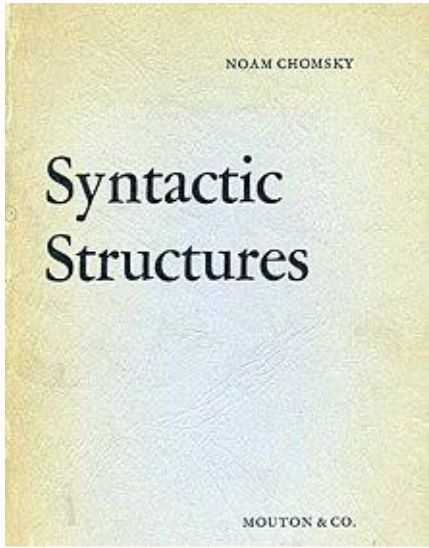
1957 Chomsky
Syntactic structures & formal grammars



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

1951 Shannon → 1957 Chomsky → 1967 Gold

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		1957 Chomsky		1967 Gold
Prediction and entropy of English	→	*Syntactic structures & formal grammars*	→	*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: UBS and Yahoo Finance

Modern Language Generators – LLMs

Source: UBS and Yahoo Finance

Modern Language Generators – LLMs

Source: UBS and Yahoo Finance

Modern Language Generators – LLMs

Source: UBS and Yahoo Finance

Modern Language Generators – LLMs

Source: UBS and Yahoo Finance

Modern Language Generators – LLMs

Source: UBS and Yahoo Finance

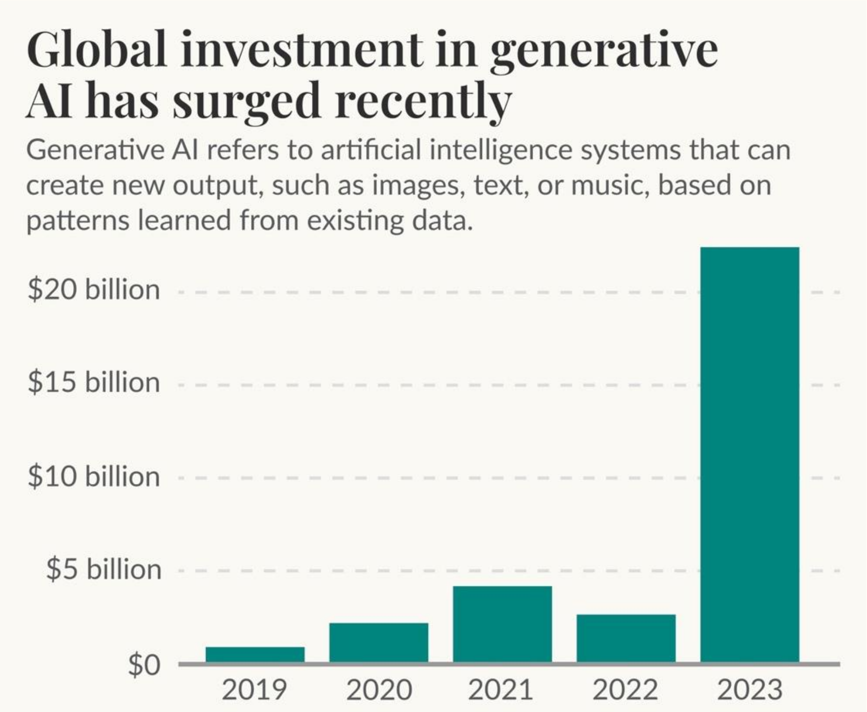
Modern Language Generators – LLMs

Source: UBS and Yahoo Finance

Modern Language Generators – LLMs

Source: UBS and Yahoo Finance

Modern Language Generators – LLMs



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

Hallucinations in LLMs

Give a brief definition of hallucinations in language models.

Reasoned about hallucinations in language models for a few seconds >

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.

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.

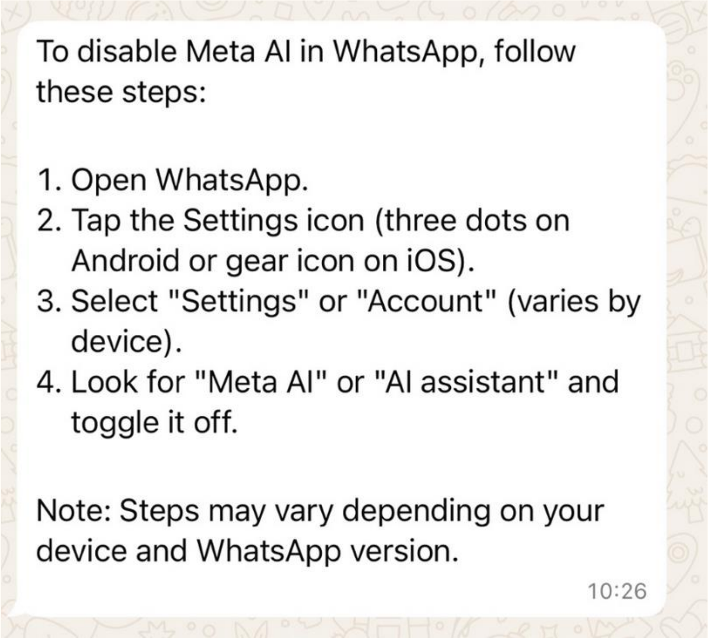
Subsequently … hallucinations in products

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 are more rare on "common tasks"

- Due to a variety of techniques:
  → chain-of-thought,
  → auxiliary tools (e.g., web search), …
- Models still hallucinate and make errors on more complex tasks (e.g., proofs, real world 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 are more rare on "common tasks"

- Due to a variety of techniques:
  → chain-of-thought,
  → auxiliary tools (e.g., web search), …
- Models still hallucinate and make errors on more complex tasks (e.g., proofs, real world tasks, …)

Question. Can hallucinations be avoided by better (but "*similar*") models/training or is fundamental change needed?

Outline of the Talk

1. Motivation: CS and Language Generation

2. Model

3. Overview Our Definitions and Results
   a. Characterizations I (Generation with Breadth)
   b. Characterizations II (*Stable* Generation with Breadth)
   c. Beyond Characterizations
   d. Learning Curves

4. Overview of *Some* Proofs

A Model of Language Generation

We introduce the model of language generation by Kleinberg and Mullainathan (2024), which builds on [Gold'67] and [Angluin'79]

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

The *Core* Task in Language Generation

Language Generation [Kleinberg-Mullainathan, 2024]
Given text from unknown language, produce *valid* and *unseen* text

The *Core* Task in Language Generation

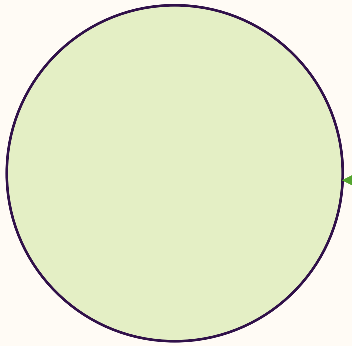
Language Generation [Kleinberg-Mullainathan, 2024]
Given text from unknown language, produce *valid* and *unseen* text

Simplifications
- Do not need to learn *all* of the language, okay to learn a subset
- It is a prompt-less model – can be extended

The *Core* Task in Language Generation

Language Generation [Kleinberg-Mullainathan, 2024]
Given text from unknown language, produce *valid* and *unseen* text

Simplifications
- Do not need to learn *all* of the language, okay to learn a subset
- It is a prompt-less model – can be extended



Target language *K*

The *Core* Task in Language Generation

Language Generation [Kleinberg-Mullainathan, 2024]
Given text from unknown language, produce *valid* and *unseen* text

Simplifications
- Do not need to learn *all* of the language, okay to learn a subset
- It is a prompt-less model – can be extended

The *Core* Task in Language Generation

Language Generation [Kleinberg-Mullainathan, 2024]
Given text from unknown language, produce *valid* and *unseen* text

Simplifications

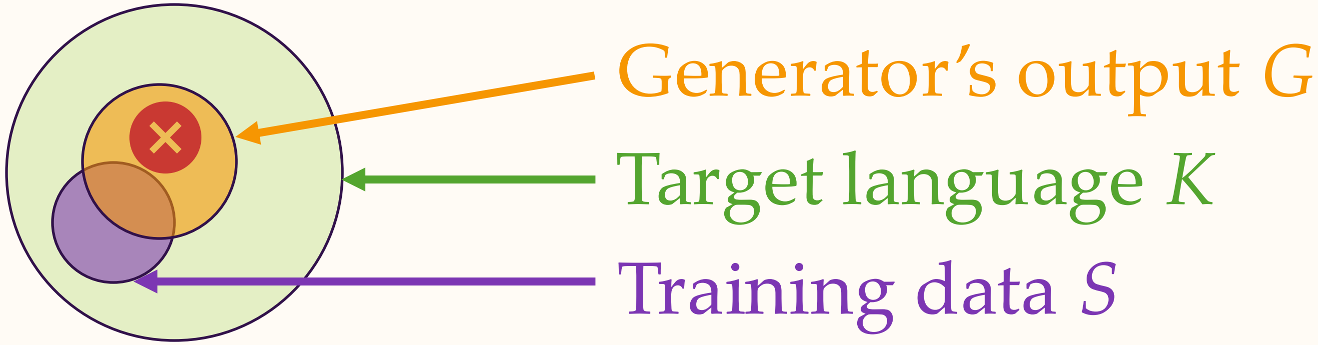
- Do not need to learn *all* of the language, okay to learn a subset
- It is a prompt-less model – can be extended

The *Core* Task in Language Generation

Language Generation [Kleinberg-Mullainathan, 2024]
Given text from unknown language, produce *valid* and *unseen* text

Simplifications

- Do not need to learn *all* of the language, okay to learn a subset
- It is a prompt-less model – can be extended



The *Core* Task in Language Generation

Language Generation [Kleinberg-Mullainathan, 2024]
Given text from unknown language, produce *valid* and *unseen* text

Simplifications
- Do not need to learn *all* of the language, okay to learn a subset
- It is a prompt-less model – can be extended

Model of Language Identification

Language Identification in the Limit [Gold, 1967]

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

Model of Language Identification

Language Identification in the Limit [Gold, 1967]

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

E.g., regular languages

Model of Language Identification

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 $\boldsymbol{K} = \boldsymbol{L_{i^\star}}$

Model of Language Identification

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

Model of Language Identification

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: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \ldots$

Model of Language Identification

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: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{\ldots}$

Model of Language Identification

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: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \ldots$

Adversary has to present a complete enumeration

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

Model of Language Identification

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: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \ldots$

Adversary has to present a complete enumeration

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

Model of Language Identification

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

Model of Language Identification

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: $\ldots, i_t, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{i^\star}, \boldsymbol{\ldots}$

Adversary has to present a complete enumeration

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

Model of Language Identification

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:

Model of Language Identification

Language Identification in the Limit [Gold, 1967]

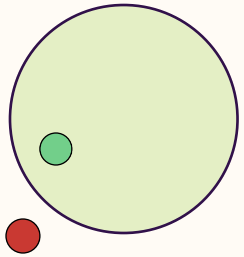
Game between adversary and learner

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



Model of Language Identification

Language Identification in the Limit [Gold, 1967]

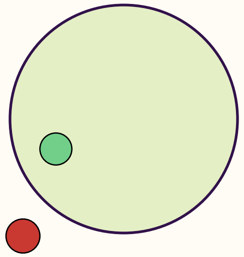
Game between adversary and learner

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:

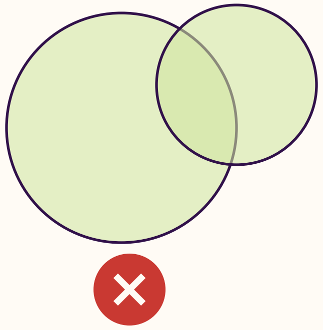
Is $w \in L_i$?

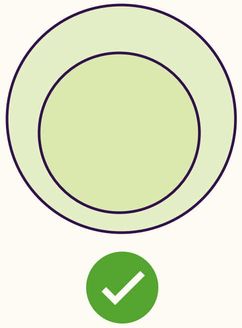
Membership Query



Is $L_i \subseteq L_j$?

Subset Query





Model of Language Generation

Language Generation in the Limit [Kleinberg-Mullainathan'24]

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

Model of Language Generation

Language Generation in the Limit [Kleinberg-Mullainathan'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 $K = L_{i^\star}$

Model of Language Generation

Language Generation in the Limit [Kleinberg-Mullainathan'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

Model of Language Generation

Language Generation in the Limit [Kleinberg-Mullainathan'24]

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

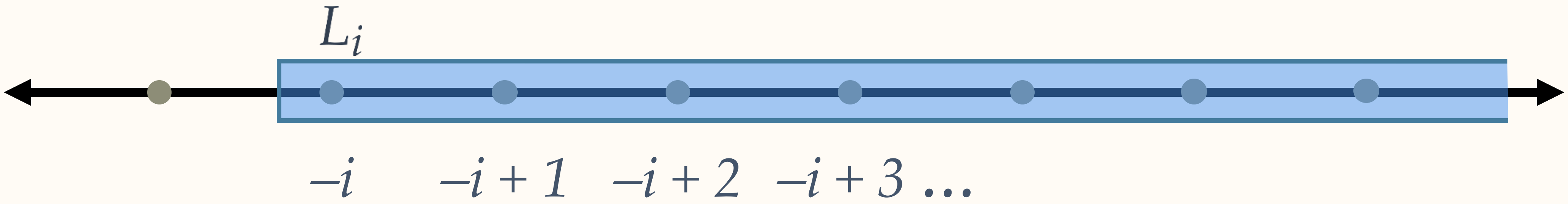
Game between adversary and generator

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 $t < \infty$

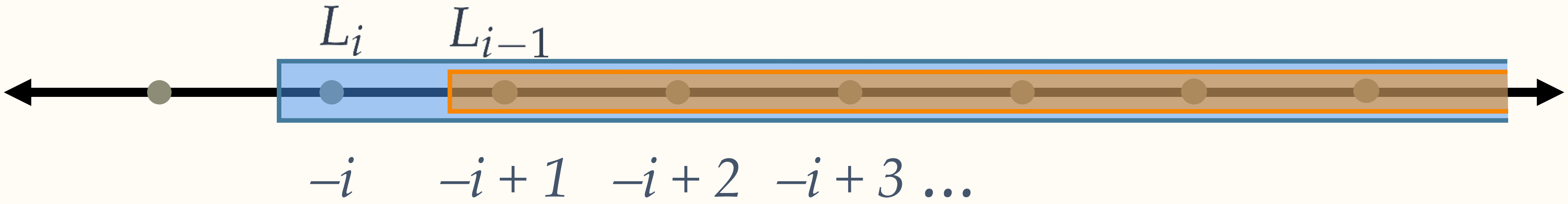
Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

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



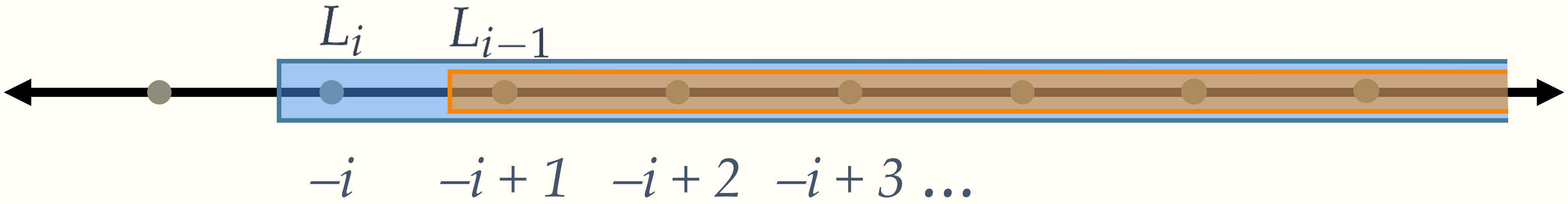
Example [Kleinberg-Mullainathan' 24] [Charikar-Pabbaraju'24]

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



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

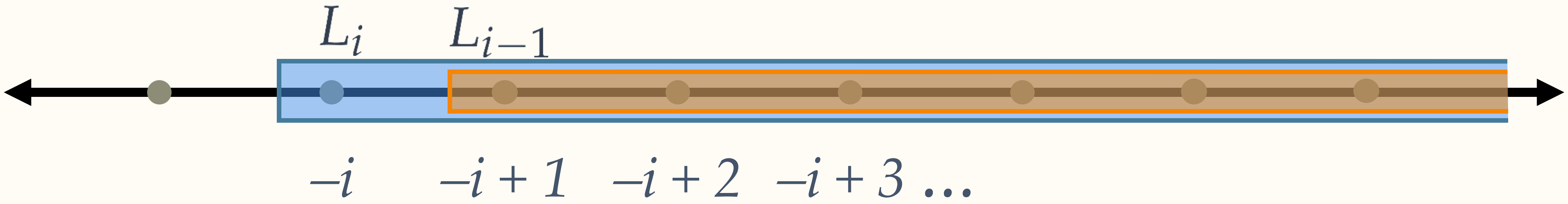
$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\,.$$



▷ Is $\mathcal{L}$ generatable?

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

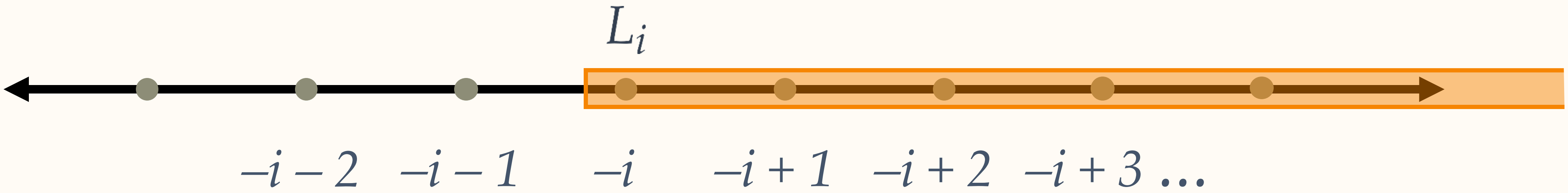
$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \dots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \dots\}\,.$$



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

Example [KM'24] [CP'24]

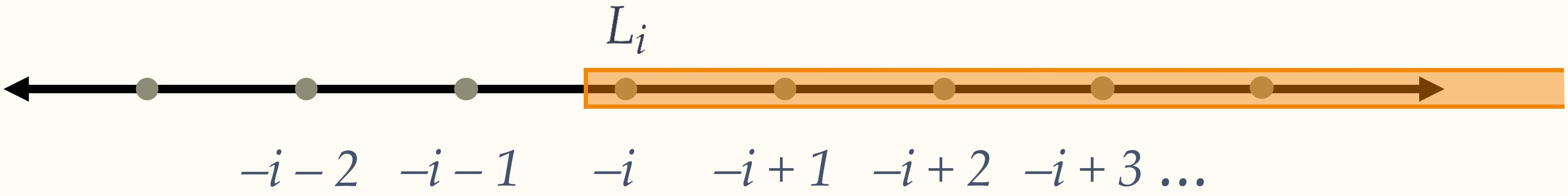
$$\mathcal{L} = \{\mathbb{Z}, L_1, L_2, \ldots\} \quad \text{where} \quad L_i = \{-i, -i+1, -i+2, \ldots\}\,.$$



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

Example [KM'24] [CP'24]

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

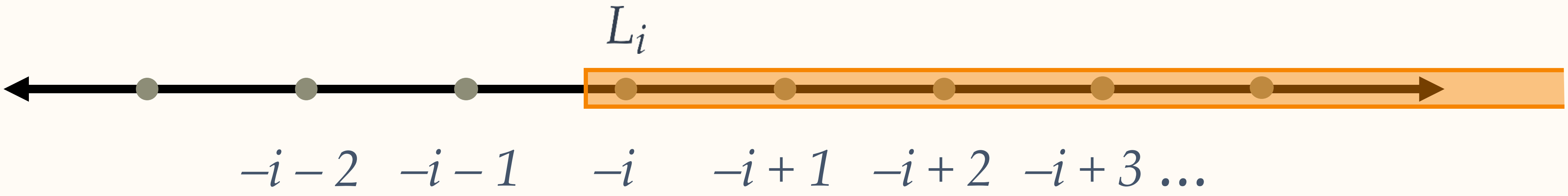


▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \ldots\}$

▷ Is $\mathcal{L}$ identifiable?

Example [KM'24] [CP'24]

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



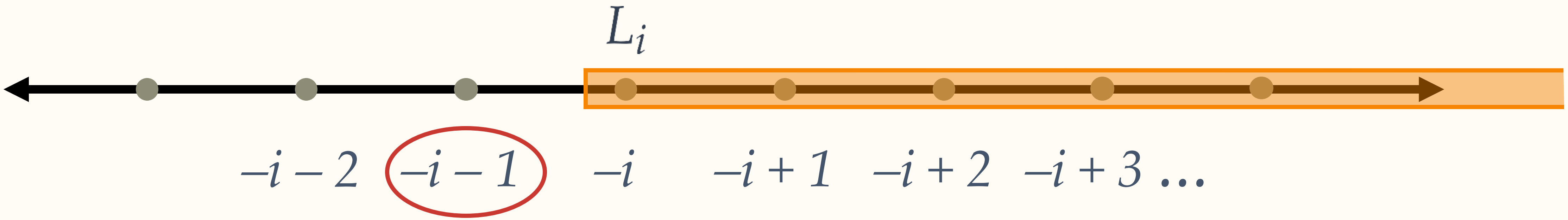
▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1 + 1, x_2 + 2, \ldots\}$

▷ Is $\mathcal{L}$ identifiable?

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

Example [KM'24] [CP'24]

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



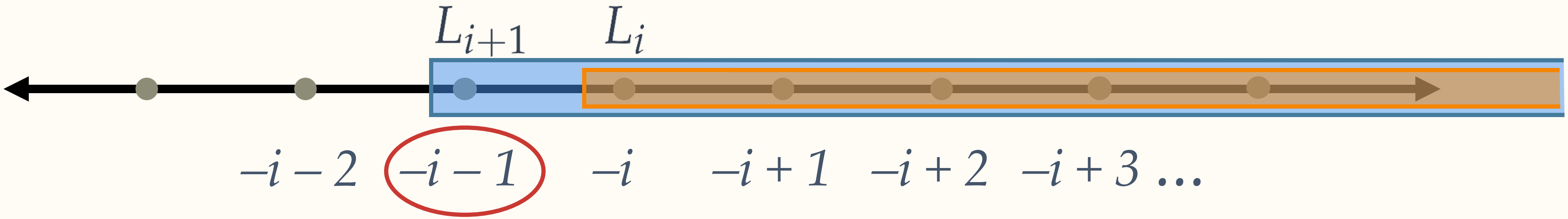
▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \ldots\}$

▷ Is $\mathcal{L}$ identifiable?

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

Example [KM'24] [CP'24]

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



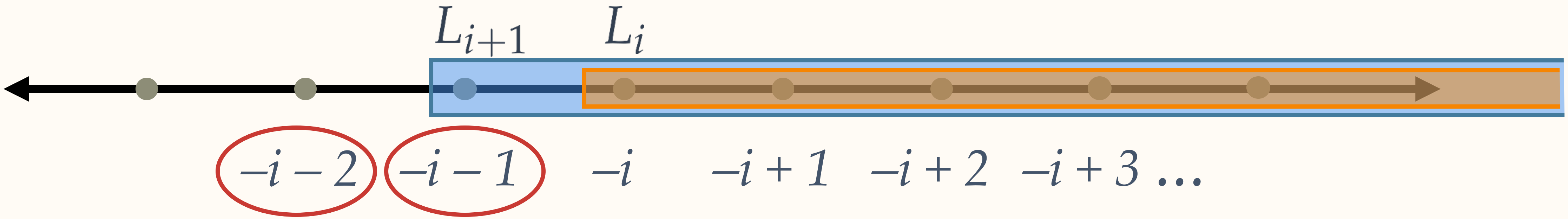
▷ Is $\mathcal{L}$ generatable? Yes, even with a single sample!
Output an unseen example from $\{x_1+1, x_2+2, \ldots\}$

▷ Is $\mathcal{L}$ identifiable?

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

Example [KM'24] [CP'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

Model of Language Generation

Language Generation [KM'24]. Generator wins if guesses are eventually in K: $\boldsymbol{K \ni g_t, g_{t+1}, \ldots}$ after some finite time $t < \infty$

Model of Language Generation

Language Generation [KM'24]. Generator wins if guesses are eventually in K: $K \ni g_t, g_{t+1}, \ldots$ after some finite time $t < \infty$

Abstractly captures many aspects of LLM training 

- *No feedback* for generator
- Generator tries to generate *unseen strings*
- Generator *cannot* ask if $w \in K$

Model of Language Generation

Language Generation [KM'24]. Generator wins if guesses are eventually in K: $K \ni g_t, g_{t+1}, \ldots$ after some finite time $t < \infty$

Abstractly captures many aspects of LLM training 

- *No feedback* for generator
- Generator tries to generate *unseen strings*
- Generator *cannot* ask if $w \in K$

Details abstracted away: computation, next-token-prediction, …

These are important …[Bhattamishra, Ahuja, and Goyal'20] [Sanford, Hsu, Telgarsky'23] [Peng, Narayanan, and Papadimitriou'24] [Chen, Peng, and Wu'24]…

Model of Language Generation

Language Generation [KM'24]. Generator wins if guesses are eventually in K: $K \ni g_t, g_{t+1}, \ldots$ after some finite time $t < \infty$

Abstractly captures many aspects of LLM training 

- *No feedback* for generator
- Generator tries to generate *unseen strings*
- Generator *cannot* ask if $w \in K$

Question. *Even in an idealized model,* can hallucinations be avoided with better models/training or is fundamental change needed?

Is Language Generation Feasible?

Informal Theorem [Gold'67, Angluin'79, '80] Almost all interesting language collections $\mathcal{L}$ are *not* identifiable

Even regular languages are non-identifiable… simpler than English

Is Language Generation Feasible?

Informal Theorem [Gold'67, Angluin'79, '80] Almost all interesting language collections $\mathcal{L}$ are *not* identifiable

Even regular languages are non-identifiable… simpler than English

Informal Theorem [Kleinberg-Mullainathan'24] All (countable) language collections $\mathcal{L}$ *are* generatable

Outline of the Talk

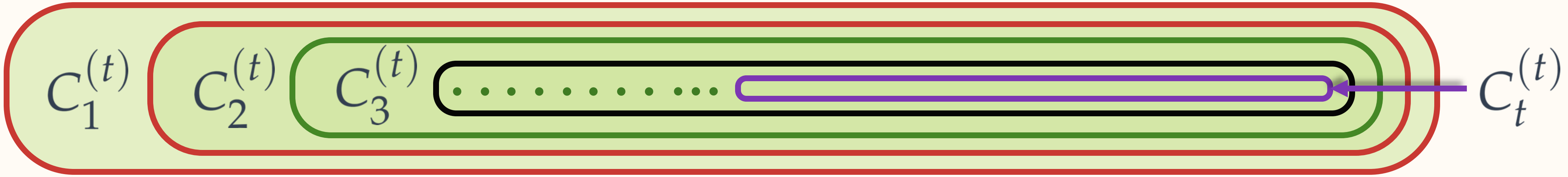
1. Motivation: CS and Language Generation
2. Model
3. Overview Our Definitions and Results
    a. Characterizations I (Generation with Breadth)
    b. Characterizations II (*Stable* Generation with Breadth)
    c. Beyond Characterizations
    d. Learning Curves
4. Overview of *Some* Proofs

[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$

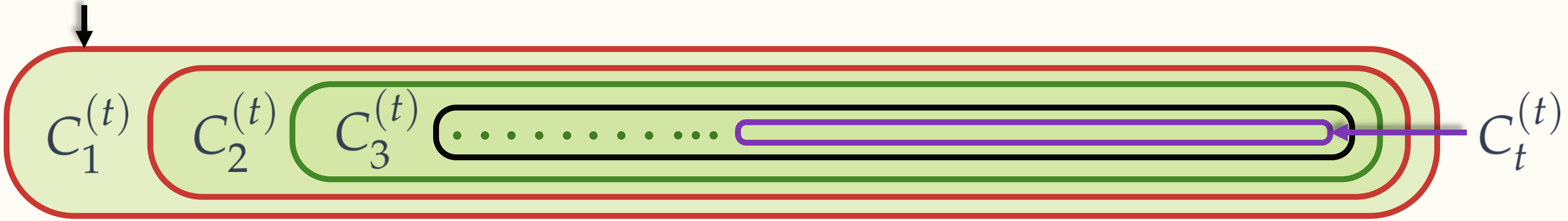


[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$

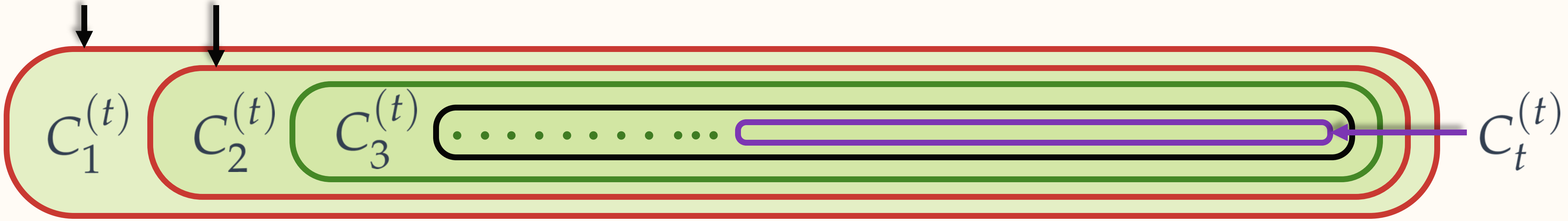


[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$

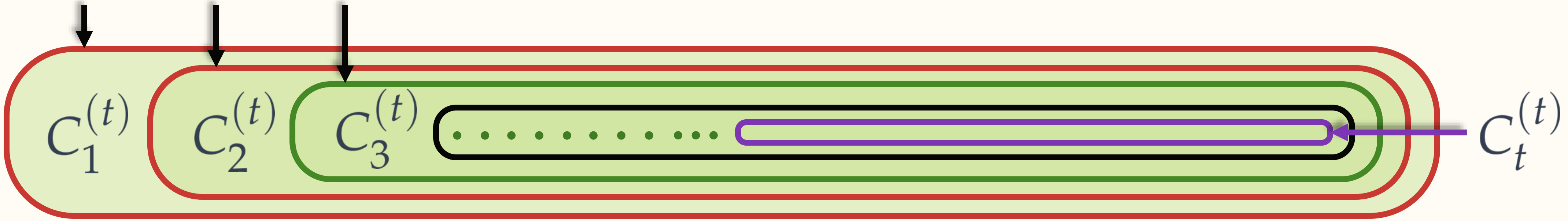


[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \ldots$$

▷ In the t-th step, generate from $C_t^{(t)}$

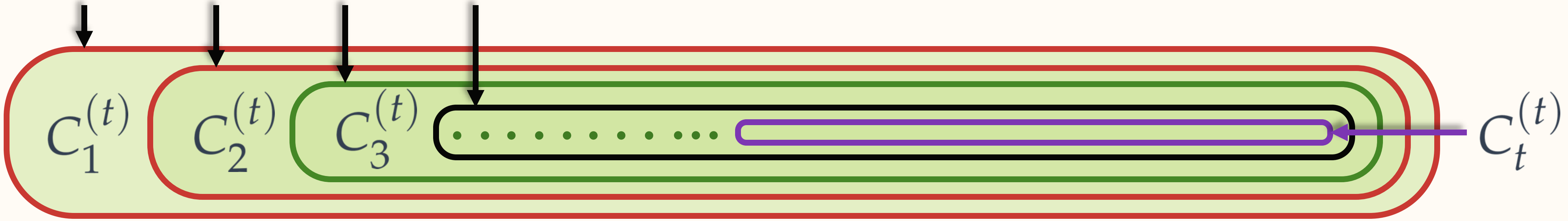


[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$

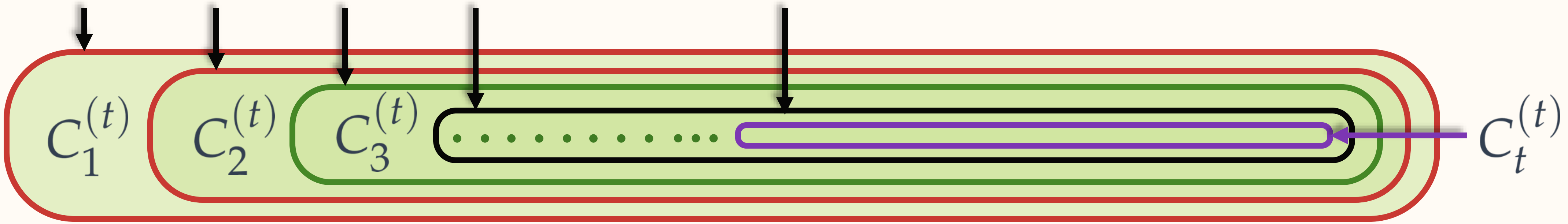


[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$



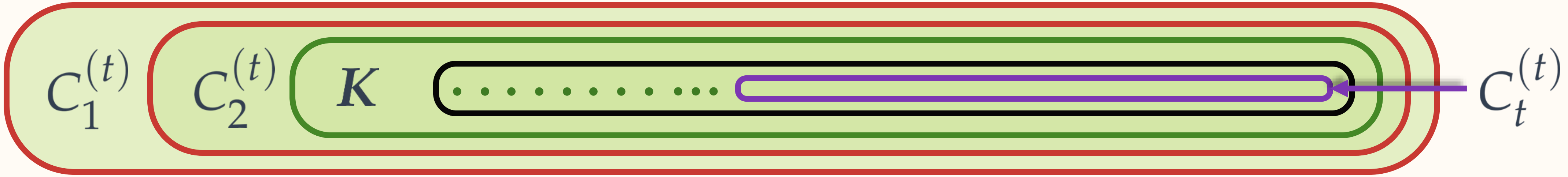
[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$

Lemma. K enters the *critical list* at finite $t_K < \infty$ and never leaves



[KM'24]'s Generator Loses Breadth

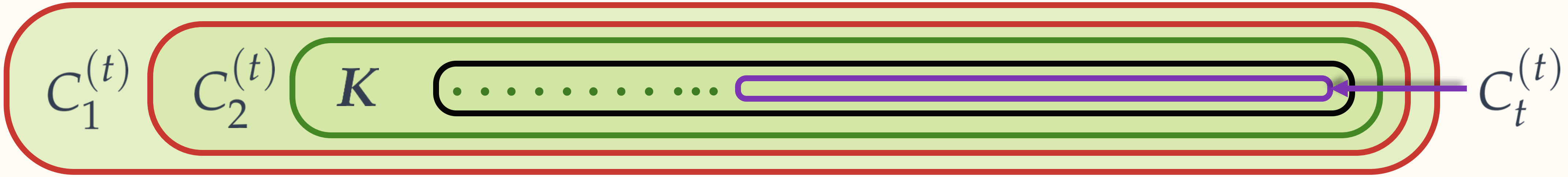
[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$

Lemma. K enters the *critical list* at finite $t_K < \infty$ and never leaves

▷ For $t > t_K$, generator outputs from $C_t^{(t)} \subsetneq K$ (no hallucinations!)



[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \dots$$

▷ In the t-th step, generate from $C_t^{(t)}$

Lemma. K enters the *critical list* at finite $t_K < \infty$ and never leaves

▷ For $t > t_K$, generator outputs from $C_t^{(t)} \subsetneq K$ (no hallucinations!)

▷ [KM'24]'s algorithm generates from a "decreasing" subset of K

[KM'24]'s Generator Loses Breadth

[KM'24]'s **Generator:** Construct (dynamic) list of *critical languages*

$$C_1^{(t)} \supsetneq C_2^{(t)} \supsetneq \cdots \supseteq C_i^{(t)} \supsetneq \cdots$$

▷ In the t-th step, generate from $C_t^{(t)}$

Lemma. K enters the *critical list* at finite $t_K < \infty$ and never leaves

▷ For $t > t_K$, generator outputs from $C_t^{(t)} \subsetneq K$ (no hallucinations!)

▷ [KM'24]'s algorithm generates from a "decreasing" subset of K

Adaptation of our main question [KM24]: Can a generator avoid hallucinations while maintaining some notion of "breadth"?

Language Generation

Characterizations I

- *We introduce several notions of breadth*
- *We show that achieving breadth + no hallucinations is impossible for most language collections*

Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Exact Breadth

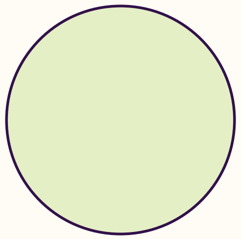
$$G(S) = K$$

Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Exact Breadth

$G(S) = K$

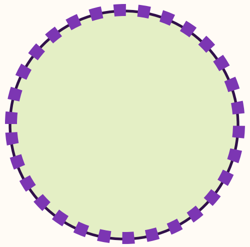


Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Exact Breadth

$G(S) = K$



Notions of Coverage or Breadth

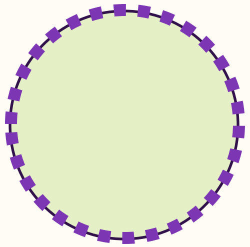
Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Exact Breadth

$$G(S) = K$$

Approximate Breadth

$$|K \setminus G(S)| < \infty$$

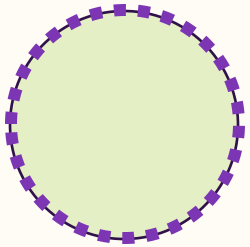


Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

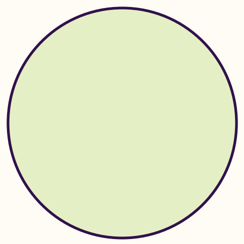
Exact Breadth

$$G(S) = K$$



Approximate Breadth

$$|K \setminus G(S)| < \infty$$

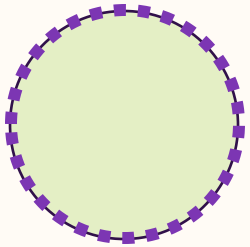


Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

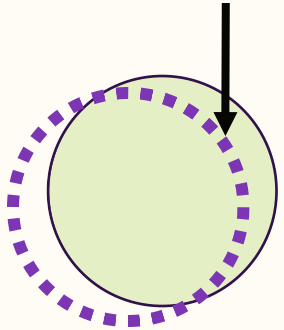
Exact Breadth

$$G(S) = K$$



Approximate Breadth

$$|K \setminus G(S)| < \infty$$



Notions of Coverage or Breadth

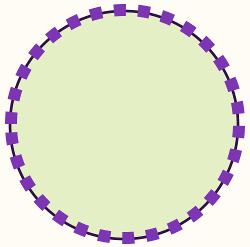
Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

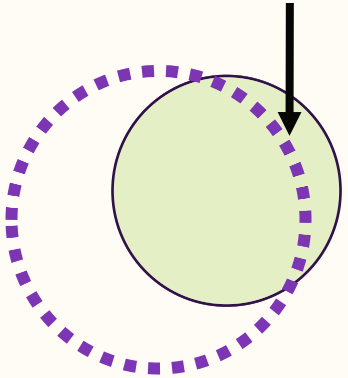
Exact Breadth

$G(S) = K$

Approximate Breadth

$|K \setminus G(S)| < \infty$





Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Exact Breadth

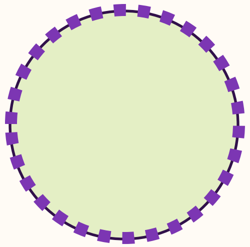
$G(S) = K$

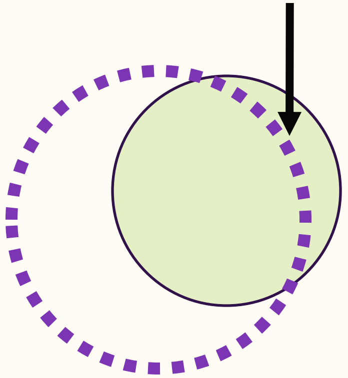
Approximate Breadth

$|K \setminus G(S)| < \infty$

Infinite Coverage

$|K \cap G(S)| = \infty$





Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Exact Breadth

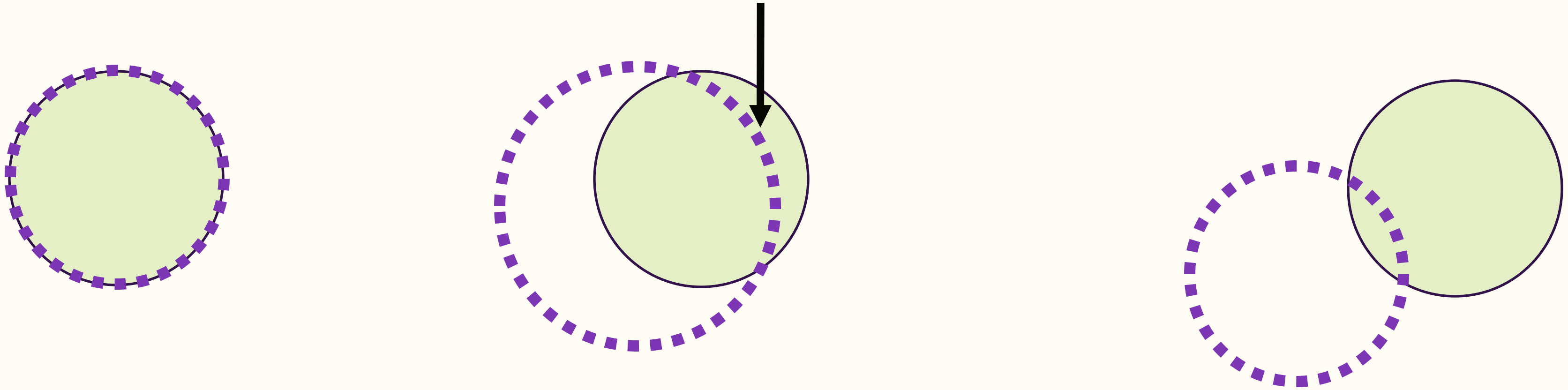
$G(S) = K$

Approximate Breadth

$|K \setminus G(S)| < \infty$

Infinite Coverage

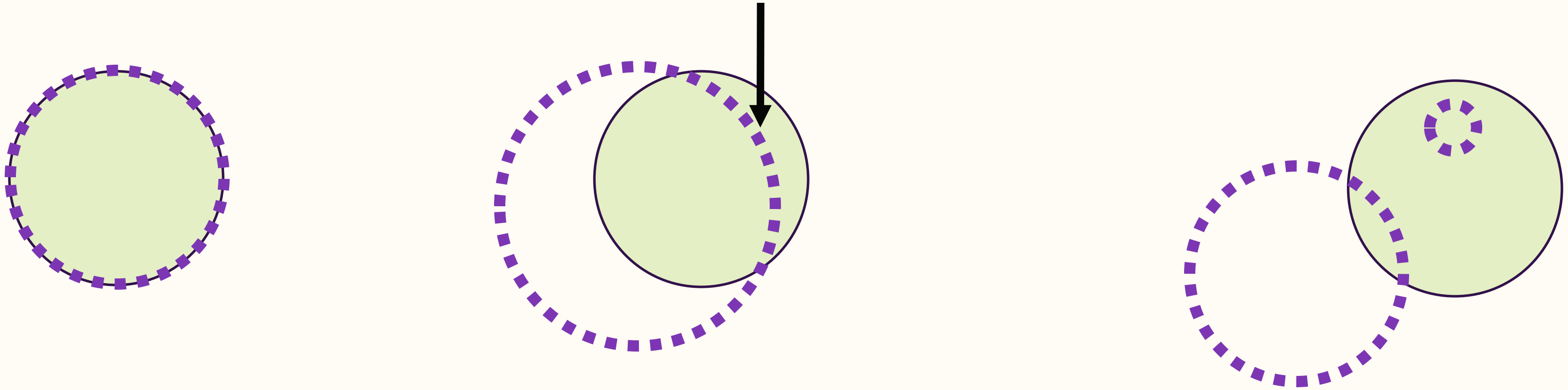
$|K \cap G(S)| = \infty$



Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

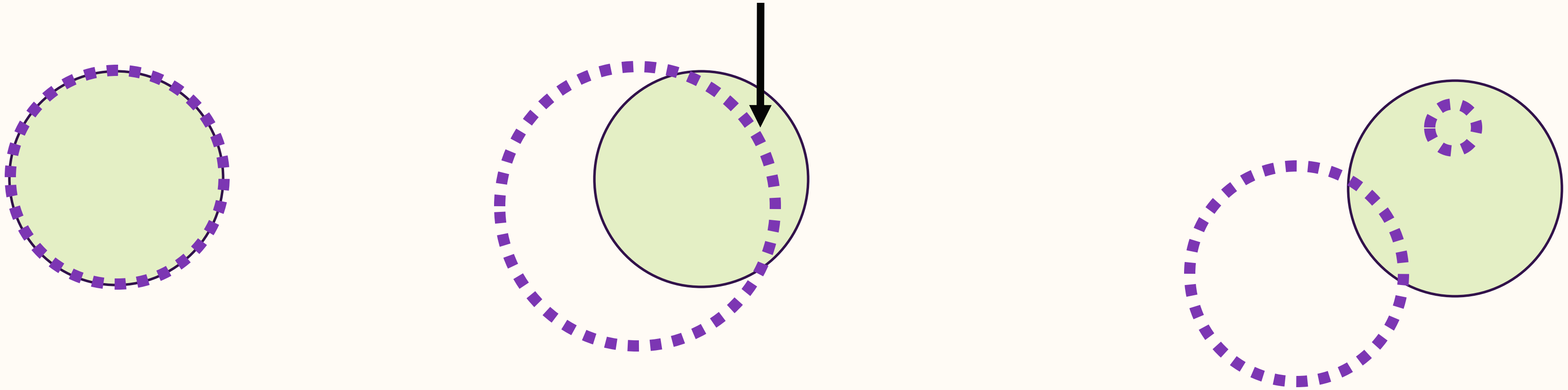
Exact Breadth	**Approximate Breadth**	**Infinite Coverage**
$G(S) = K$	$\lvert K \setminus G(S) \rvert < \infty$	$\lvert K \cap G(S) \rvert = \infty$



Notions of Coverage or Breadth

Let generator G be a mapping from training data to subsets of the domain, *i.e.*, $G(S)$ is output-set of G trained on S

Exact Breadth	**Approximate Breadth**	**Infinite Coverage**
$G(S) = K$	$\lvert K \setminus G(S) \rvert < \infty$	$\lvert K \cap G(S) \rvert = \infty$



Consider $K = \mathbb{N}$, $\quad G(S) = \{i, i+1, \dots\} \quad$ and $\quad G(S) = \{2, 4, 6, \dots\}$

Results for Breadth with *no* Hallucination

Results for Breadth with *no* Hallucination

Infinite coverage $\iff$ Generation $\iff$ All countable collections

[KM'24]

Results for Breadth with *no* Hallucination

Infinite coverage $\iff$ Generation $\iff$ All countable collections
[KM'24]

Exact Breadth $\iff$ Angluin's Condition [Angluin,1980]

Results for Breadth with *no* Hallucination

Infinite coverage $\iff$ Generation $\iff$ All countable collections
[KM'24]

Approximate Breadth $\iff$ Weak Angluin's Condition [KMV24]

Exact Breadth $\iff$ Angluin's Condition [Angluin,1980]

Results for Breadth with *no* Hallucination

Infinite coverage $\iff$ Generation $\iff$ All countable collections
[KM'24]

Not satisfied even, e.g., by regular languages

Approximate Breadth $\iff$ Weak Angluin's Condition [KMV24]

Exact Breadth $\iff$ Angluin's Condition [Angluin,1980]

Results for Breadth with *no* Hallucination

Infinite coverage $\iff$ Generation $\iff$ All countable collections
[KM'24]

Not satisfied even, e.g., by regular languages

Approximate Breadth $\iff$ Weak Angluin's Condition [KMV24]

Exact Breadth $\iff$ Angluin's Condition [Angluin,1980]

Main Takeaway [Kalavasis, **M.**, Velegkas'24 and '25]**.** For most interesting language collections, LLMs cannot avoid hallucination while achieving any of these notions of breadth

Technical Vignette: Properties of Breadth

Definitions. A relation P satisfies:

- Uniqueness if $(\mathcal{G}, L), (\mathcal{G}, L') \in P$, then $L = L'$
- Finite non-uniqueness if $(\mathcal{G}, L), (\mathcal{G}, L') \in P$, then $|L \triangle L'| < \infty$

Technical Vignette: Properties of Breadth

Definitions. A relation P satisfies:

- ▷ Uniqueness if $(\mathcal{G}, L), (\mathcal{G}, L') \in P$, then $L = L'$
- ▷ Finite non-uniqueness if $(\mathcal{G}, L), (\mathcal{G}, L') \in P$, then $|L \triangle L'| < \infty$

Fact. Exact breadth satisfies uniqueness

Fact. Approximate breadth satisfies finite non-uniqueness

Technical Vignette: Properties of Breadth

Definitions. A relation P satisfies:

- ▷ Uniqueness if $(\mathcal{G}, L), (\mathcal{G}, L') \in P$, then $L = L'$
- ▷ Finite non-uniqueness if $(\mathcal{G}, L), (\mathcal{G}, L') \in P$, then $|L \triangle L'| < \infty$

Fact. Exact breadth satisfies uniqueness

Fact. Approximate breadth satisfies finite non-uniqueness

Theorem. [Kalavasis, **M.**, Velegkas, 24] Consider collection $\mathcal{L}$.

- ▷ If P is unique and $\mathcal{L}$ violates Angluin's condition, or
- ▷ If P is finite-non-unqiue and $\mathcal{L}$ violates weak Angluin's condition,

Then, no generator can satisfy P in the limit

Results for Breadth with *some* Hallucination

	No Hallucinations $\lvert G(S_t)\backslash K\rvert = 0$	**Finite Hallucinations** $\lvert G(S_t)\backslash K\rvert < \infty$	**Infinite Hallucinations** $\lvert G(S_t)\backslash K\rvert = \infty$
Zero Missing Elements $\lvert K\backslash G(S_t)\rvert = 0$	Angluin's Condition [Ang 80] *(i.e., Exact Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Finite Missing Elements $\lvert K\backslash G(S_t)\rvert < \infty$	Weak Angluin's Condition [KMV 24b, CP 24] *(i.e., Approximate Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Infinite Present Elements $\lvert K \cap G(S_t)\rvert = \infty$	All Countable Collections	All Countable Collections	All Countable Collections

Results for Breadth with *some* Hallucination

	No Hallucinations $\lvert G(S_t) \setminus K \rvert = 0$	**Finite Hallucinations** $\lvert G(S_t) \setminus K \rvert < \infty$	**Infinite Hallucinations** $\lvert G(S_t) \setminus K \rvert = \infty$
Zero Missing Elements $\lvert K \setminus G(S_t) \rvert = 0$	Angluin's Condition [Ang 80] *(i.e., Exact Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Finite Missing Elements $\lvert K \setminus G(S_t) \rvert < \infty$	Weak Angluin's Condition [KMV 24b, CP 24] *(i.e., Approximate Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Infinite Present Elements $\lvert K \cap G(S_t) \rvert = \infty$	All Countable Collections	All Countable Collections	All Countable Collections

Question. Can one develop a more fine-grained characterization?

Results for Breadth with *some* Hallucination

	No Hallucinations $\lvert G(S_t)\setminus K\rvert = 0$	**Finite Hallucinations** $\lvert G(S_t)\setminus K\rvert < \infty$	**Infinite Hallucinations** $\lvert G(S_t)\setminus K\rvert = \infty$
Zero Missing Elements $\lvert K\setminus G(S_t)\rvert = 0$	Angluin's Condition [Ang 80] *(i.e., Exact Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Finite Missing Elements $\lvert K\setminus G(S_t)\rvert < \infty$	Weak Angluin's Condition [KMV 24b, CP 24] *(i.e., Approximate Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Infinite Present Elements $\lvert K\cap G(S_t)\rvert = \infty$	All Countable Collections	All Countable Collections	All Countable Collections

Question. Can one develop a more fine-grained characterization?

Progress. [Kleinberg and Wei, 2025] and [Peale, Raman, Reingold, 2025]

Outline of the Talk

1. Motivation: CS and Language Generation
2. Model
3. Overview Our Definitions and Results
    a. Characterizations I (Generation with Breadth)
    b. Characterizations II (*Stable* Generation with Breadth)
    c. Beyond Characterizations
    d. Learning Curves
4. Overview of *Some* Proofs

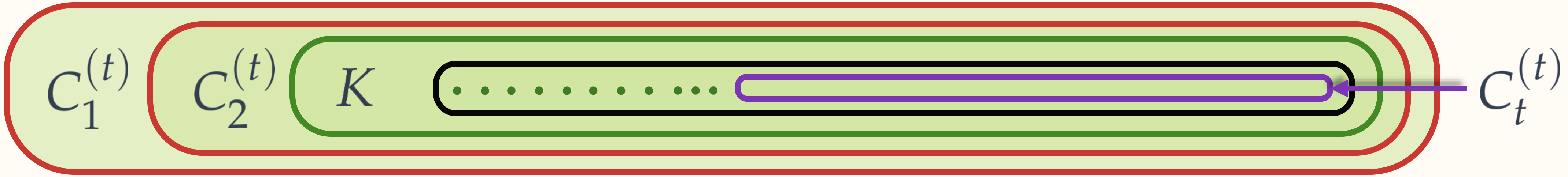
Language Generation

Characterizations II

- *We introduce stable generation*
- *Related to whether the learner can recognize that it has learnt?*
- *Requiring stability makes language generation much harder*

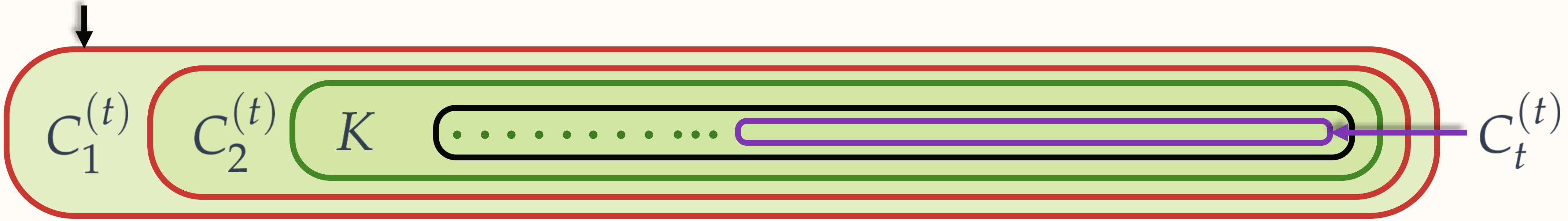
Stability in Language Generation

[KM'24]'s and our generator change output set infinitely often



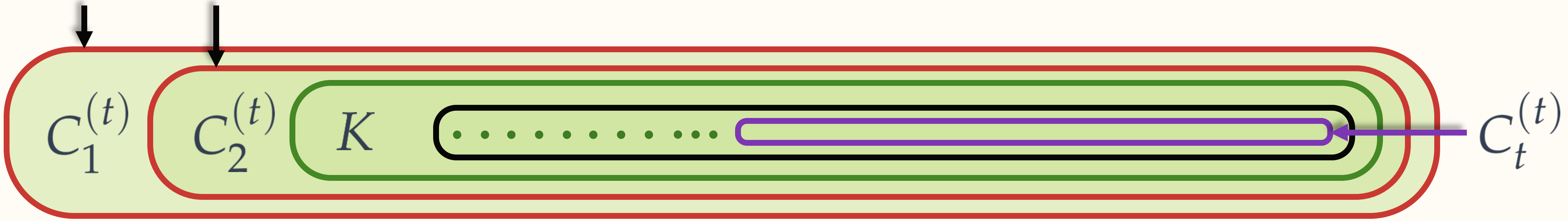
Stability in Language Generation

[KM'24]'s and our generator change output set infinitely often



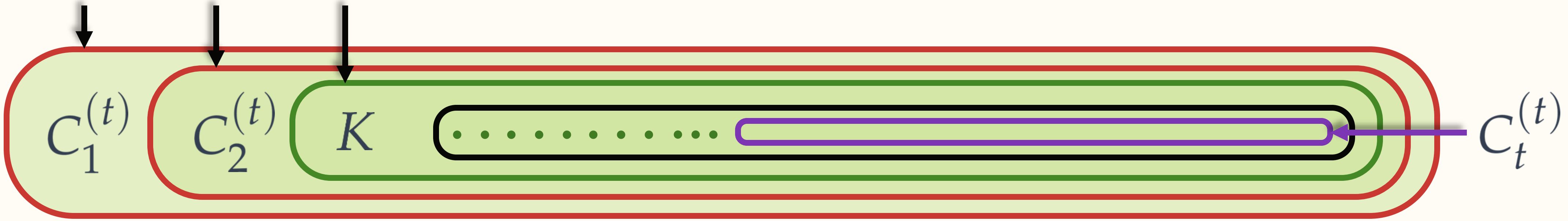
Stability in Language Generation

[KM'24]'s and our generator change output set infinitely often



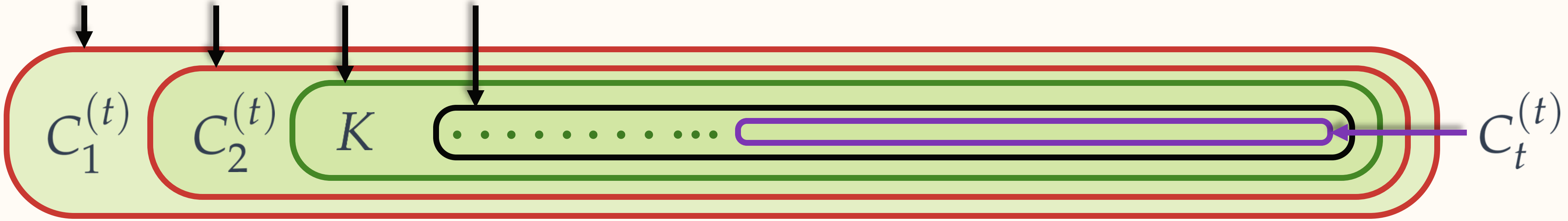
Stability in Language Generation

[KM'24]'s and our generator change output set infinitely often



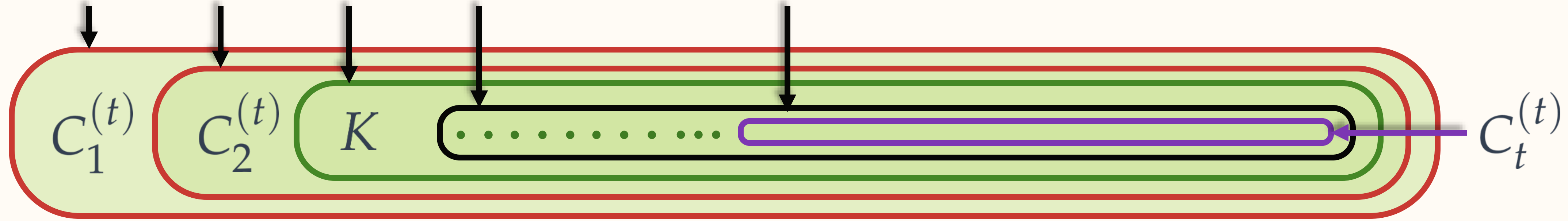
Stability in Language Generation

[KM'24]'s and our generator change output set infinitely often



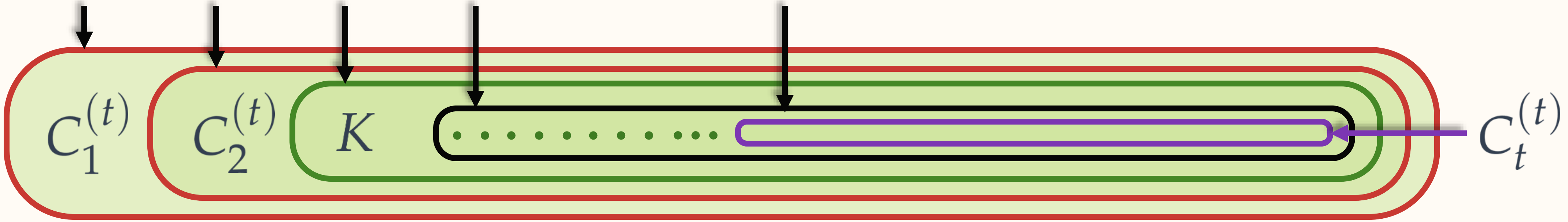
Stability in Language Generation

[KM'24]'s and our generator change output set infinitely often



Stability in Language Generation

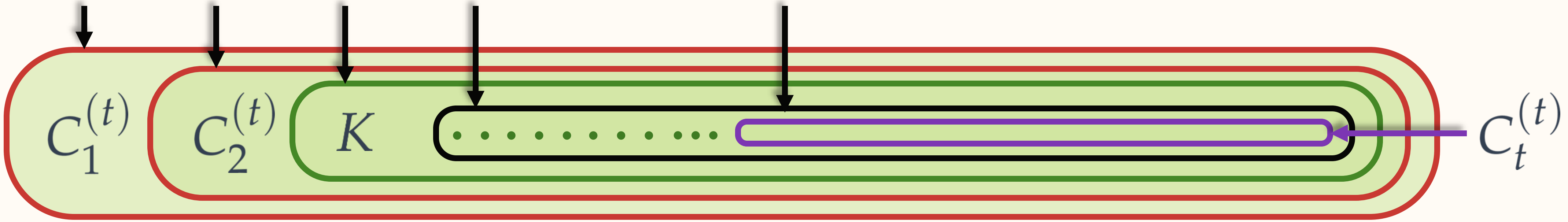
[KM'24]'s and our generator change output set infinitely often



Can generator's stabilize their outputs?
If the generator "knows" it has learnt, then it can stabilize.

Stability in Language Generation

[KM'24]'s and our generator change output set infinitely often



Can generator's stabilize their outputs?
If the generator "knows" it has learnt, then it can stabilize.

Stability. A generator is said to achieve *stability* if for any target K and its enumeration, there is a finite $t < \infty$, after which $G(S_t) = G(S_{t'})$ for all $t' \geq t$

Results with Stability

	No Hallucinations $\lvert G(S_t) \setminus K \rvert = 0$	**Finite Hallucinations** $\lvert G(S_t) \setminus K \rvert < \infty$	**Infinite Hallucinations** $\lvert G(S_t) \setminus K \rvert = \infty$
Zero Missing Elements $\lvert K \setminus G(S_t) \rvert = 0$	Angluin's Condition [Ang 80] *(i.e., Exact Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Finite Missing Elements $\lvert K \setminus G(S_t) \rvert < \infty$	Weak Angluin's Condition [KMV 24b, CP 24] *(i.e., Approximate Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Infinite Present Elements $\lvert K \cap G(S_t) \rvert = \infty$	All Countable Collections	All Countable Collections	All Countable Collections

Results with Stability

	No Hallucinations $\lvert G(S_t) \setminus K \rvert = 0$	**Finite Hallucinations** $\lvert G(S_t) \setminus K \rvert < \infty$	**Infinite Hallucinations** $\lvert G(S_t) \setminus K \rvert = \infty$
Zero Missing Elements $\lvert K \setminus G(S_t) \rvert = 0$	Angluin's Condition [Ang 80] *(i.e., Exact Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Finite Missing Elements $\lvert K \setminus G(S_t) \rvert < \infty$	Angluin's Condition [Ang 80] *(i.e., Approximate Breadth)*	Weak Angluin's Condition [KMV 24b, CP 24]	All Countable Collections
Infinite Present Elements $\lvert K \cap G(S_t) \rvert = \infty$	Characterization ? *(Not all countable collections)*	Characterization ?	All Countable Collections

Outline of the Talk

1. Motivation: CS and Language Generation
2. Model
3. Overview Our Definitions and Results
   a. Characterizations I (Generation with Breadth)
   b. Characterizations II (*Stable* Generation with Breadth)
   c. Beyond Characterizations
   d. Learning Curves
4. Overview of *Some* Proofs

Language Generation

Beyond Characterizations

Providing the generator with negative examples, enables achieving breadth without hallucinations!

Generation with Negative Examples

Informal Theorem [Gold'67][Kalavasis, **M.**, Velegkas'25] Consider a variation of language generation where adversary enumerates both elements in K and elements outside of K (*negative examples*). Then, *all countable collections $\mathcal{L}$ are generatable with exact breadth.*

Generation with Negative Examples

Informal Theorem [Gold'67][Kalavasis, **M.**, Velegkas'25] Consider a variation of language generation where adversary enumerates both elements in K and elements outside of K (*negative examples*). Then, *all countable collections $\mathcal{L}$ are generatable with exact breadth.*

Insights into LLM training

- Perhaps a principled explanation why RLHF is useful
- Does this suggest including negative information in pre-training would be useful?

Generation with Negative Examples

Proxies for negative examples have found to be useful

Neural Text ~~de~~Generation 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]

Generation with Negative Examples

Proxies for negative examples have found to be useful

NEURAL TEXT ~~DE~~GENERATION 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]

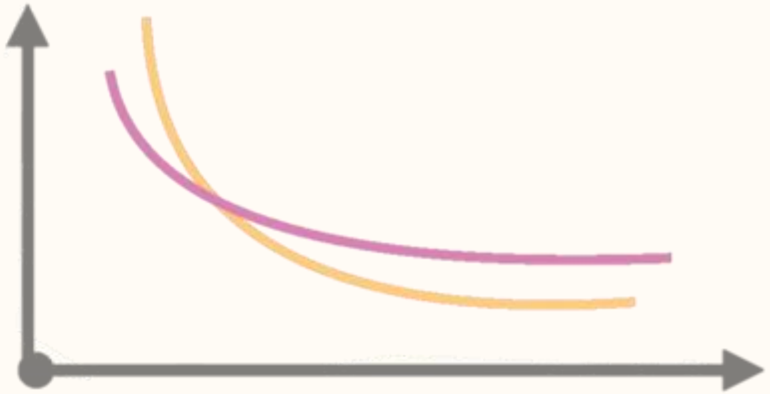
Stefano Ermon[1]

Q: *Given high and low-quality data, can one extract negative examples?*

Outline of the Talk

1. Motivation: CS and Language Generation
2. Model
3. Overview Our Definitions and Results
    a. Characterizations I (Generation with Breadth)
    b. Characterizations II (*Stable* Generation with Breadth)
    c. Beyond Characterizations
    d. Learning Curves *(How many samples are needed?)*
4. Overview of *Some* Proofs

Language Generation



Learning Curves

We establish universal rates (aka learning curves) for generation with and without breadth

Learning Curves for Generation

How many samples does one need for the generator to generate?

Learning Curves for Generation

How many samples does one need for the generator to generate?

Statistical Model of Language Generation

- Adversary picks a distribution $\mathcal{D}$ supported entirely on $K \in \mathcal{L}$
- Generator gets n i.i.d. samples from $\mathcal{D}$ and outputs $\mathcal{G}(S_n)$
- $\mathrm{Err}_n(\mathcal{G}) := \mathbb{1}\left\{\mathcal{G}(S_n) \text{ does not satisfy notion of generation}\right\}$

Learning Curves for Generation

How many samples does one need for the generator to generate?

Statistical Model of Language Generation

- Adversary picks a distribution $\mathcal{D}$ supported entirely on $K \in \mathcal{L}$
- Generator gets n i.i.d. samples from $\mathcal{D}$ and outputs $\mathcal{G}(S_n)$
- $\mathrm{Err}_n(\mathcal{G}) := \mathbb{1}\left\{\mathcal{G}(S_n) \text{ does not satisfy notion of generation}\right\}$

For fixed $\mathcal{D}$, as $n \to \infty$, how quickly does the error $\mathrm{Err}_n(\mathcal{G})$ drop?

Learning Curves for Generation

How many samples does one need for the generator to generate?

Statistical Model of Language Generation

- Adversary picks a distribution $\mathcal{D}$ supported entirely on $K \in \mathcal{L}$
- Generator gets n i.i.d. samples from $\mathcal{D}$ and outputs $\mathcal{G}(S_n)$
- $\mathrm{Err}_n(\mathcal{G}) := \mathbb{1}\{\mathcal{G}(S_n)$ does not satisfy notion of generation$\}$

For fixed $\mathcal{D}$, as $n \to \infty$, how quickly does the error $\mathrm{Err}_n(\mathcal{G})$ drop?

Informal Theorem [Kalavasis, **M.**, Velegkas'25] Error either drops exponentially quickly or is arbitrarily slow; where the characterization has tight connections to the online model

Outline of the Talk

1. Motivation: CS and Language Generation
2. Model
3. Overview Our Definitions and Results
   a. Characterizations I (Generation with Breadth)
   b. Characterizations II (*Stable* Generation with Breadth)
   c. Beyond Characterizations
   d. Learning Curves *(How many samples are needed?)*
4. Overview of *Some* Proofs

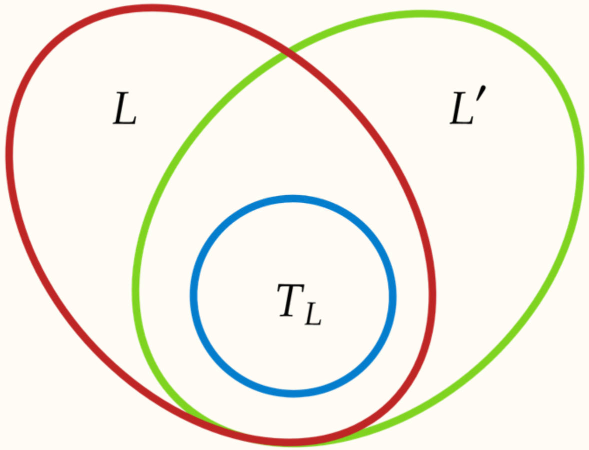
Overview of *Some* Proofs

Angluin's Condition

Definition. Language collection $\mathcal{L}$ satisfies Angluin's condition if:

For all $L \in \mathcal{L}$ there is some finite tell-tale subset $T_L \subseteq L$ such that:

For all $L' \neq L$ either $T_L \not\subseteq L'$ or L' is not a proper subset of L



Lower Bound Construction

[Kalavasis, **M.**, Velegkas'24] [Charikar, Pabbaraju'24]

Since $\mathcal{L}$ violates Angluin's condition, there is L^* such that

for all finite subsets $T \subseteq L^*$, there is $L_T \in \mathcal{L}$, $T \subseteq L_T$ and $L_T \subsetneq L^*$

Lower Bound Construction

[Kalavasis, **M.**, Velegkas'24] [Charikar, Pabbaraju'24]

Since $\mathcal{L}$ violates Angluin's condition, there is L^* such that

for all finite subsets $T \subseteq L^*$, there is $L_T \in \mathcal{L}$, $T \subseteq L_T$ and $L_T \subsetneq L^*$

1. Enumerate L^* till $\mathcal{G}$ achieves breadth (*if never, we are done*)

Lower Bound Construction

[Kalavasis, **M.**, Velegkas'24] [Charikar, Pabbaraju'24]

Since $\mathcal{L}$ violates Angluin's condition, there is L^* such that

for all finite subsets $T \subseteq L^*$, there is $L_T \in \mathcal{L}$, $T \subseteq L_T$ and $L_T \subsetneq L^*$

1. Enumerate L^* till $\mathcal{G}$ achieves breadth (*if never, we are done*)
2. Let T be the elements enumerated

Lower Bound Construction

[Kalavasis, **M.**, Velegkas'24] [Charikar, Pabbaraju'24]

Since $\mathcal{L}$ violates Angluin's condition, there is L^* such that

for all finite subsets $T \subseteq L^*$, there is $L_T \in \mathcal{L}$, $T \subseteq L_T$ and $L_T \subsetneq L^*$

1. Enumerate L^* till $\mathcal{G}$ achieves breadth (*if never, we are done*)
2. Let T be the elements enumerated
3. Continue enumerating L_T until $\mathcal{G}$ achieves breadth (*if never, we are done*) and, then, repeat from Step2

Lower Bound Construction

[Kalavasis, **M.**, Velegkas'24] [Charikar, Pabbaraju'24]

Since $\mathcal{L}$ violates Angluin's condition, there is L^* such that

for all finite subsets $T \subseteq L^*$, there is $L_T \in \mathcal{L}$, $T \subseteq L_T$ and $L_T \subsetneq L^*$

1. Enumerate L^* till $\mathcal{G}$ achieves breadth (*if never, we are done*)
2. Let T be the elements enumerated
3. Continue enumerating L_T until $\mathcal{G}$ achieves breadth (*if never, we are done*) and, then, repeat from Step2

Either Step2 repeats infinitely often and we enumerate K or we find a language on which $\mathcal{G}$ makes infinitely many mistakes.

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation
   (b) Fine-grained trade-offs between hallucinations and breadth

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation
   (b) Fine-grained trade-offs between hallucinations and breadth
2. What is the *probability* of hallucination?

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation
   (b) Fine-grained trade-offs between hallucinations and breadth
2. What is the *probability* of hallucination?
3. Allow generators to output multiple responses (could bypass many impossiblity results)

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation
   (b) Fine-grained trade-offs between hallucinations and breadth
2. What is the *probability* of hallucination?
3. Allow generators to output multiple responses (could bypass many impossiblity results)
4. Developing computationally efficient algorithms in more structured settings

Immediate Open Questions

1. Complete characterizations for the following
   (a) Stable Generation
   (b) Fine-grained trade-offs between hallucinations and breadth
2. What is the *probability* of hallucination?
3. Allow generators to output multiple responses (could bypass many impossiblity results)
4. Developing computationally efficient algorithms in more structured settings
5. Extraction of negative information from available data

Summary

1. TCS can contribute the right *abstractions* for empirical systems
    - E.g., Clustering, search, algorithmic fairness, robustness...

Summary

1. TCS can contribute the right *abstractions* for empirical systems
    - ▷ E.g., Clustering, search, algorithmic fairness, robustness...
2. We establish a tension between avoiding hallucinations while achieving breadth for existing language model "frameworks" in a theoretical model by [Kleinberg and Mullainathan'24]

Summary

1. TCS can contribute the right *abstractions* for empirical systems
   - ▷ E.g., Clustering, search, algorithmic fairness, robustness...
2. We establish a tension between avoiding hallucinations while achieving breadth for existing language model "frameworks" in a theoretical model by [Kleinberg and Mullainathan'24]
3. *How can theory guide practice?*

Thank you!

Tutorial on Language Generation

At COLT 2025, this summer!

Organized with:

Moses Charikar
Stanford

Chirag Pabbaraju
Stanford

Charlotte Peale
Stanford

Grigoris Velegkas
Yale → Google Research

Thank you!