AI-Generated Text:
Trends, Detection & Challenges

Salima Lamsiyah
University Luxembourg, FSTM, DCS MINE Research Group
Salima.lamsiyah@uni.lu

Tutorial, RANLP, Varna, 2025

Tutorial Overview

Part 1: **The Generative Revolution from n-grams to Transformers**

Part 2: **AI-Generated Text Detection Methods**

Part 3: **Benchmarks and Open Challenges**

Part 1: The Generative Revolution: From N-grams to Transformers

A Brief History of Text Generation Methods: *From N-grams to Transformers*

- Trace evolution of text generation paradigms
  - ***Trajectory:*** ***Symbolic (rule-based) -> Statistical (data-driven) -> Neural Architectures.***
- Understand limitations that led to paradigm shifts
- Deconstruct Transformer mechanics and advantages

Language Models More Formally!

- **General Goal**: compute the probability of a sentence or sequence of words:

$$\boldsymbol{Probability(W) = P(w_1, w_2, w_3, w_4, w_5 \ldots wn)}$$

- **Related Task**: predicting probability of an upcoming word:

$$\boldsymbol{Probability(w_{n-1} | w_1, w_2, w_3 \ldots, w_n)}$$

- A model that computes either of these:

  P(W) or **$P(w_n | w_1, w_2 \ldots w_{n-1})$** is called a **language model**.

Types of Language Modeling Methods

Language Models Methods

Symbolic Rule-Based Methods (1950s-1980s)

- Eliza (1964-66)
- SHRDLU (early 1970s)
- Conceptual Ontologies (MARGIE, SAM, PAM)
- …

Statistical-based Methods (1990s-2010s)

- N-grams Models
- Hidden Markov Models
- …

Neural-based Methods (2010s- 2025)

- Word2vec, GloVe
- RNN, LSTMS, GRU
- Transformer
- LLMS

Limitations of the Symbolic Era

- Required immense manual effort to encode linguistic rules and world knowledge.
- Systems were fragile, performing well only in narrow, predefined domains and failing to generalize.
- The sheer number of linguistic rules made the approach untenable for general-purpose systems.
- Led to the first "AI winter" due to the approach's limitations.

The Statistical Dawn (1990s-2010s)

- Increased availability of digital text corpora and greater computational power.
- Moved from knowledge-based to data-driven NLP.
- Instead of hand-coding rules, these models rely on statistical techniques and frequency-based approaches to predict words or sequences.
  - ***Example:*** If we often say "bread and butter," it learns to predict "butter" after "bread and."
- **Key Models:** N-gram Models and Hidden Markov Models (HMMs).

N-gram Models

- **Concept:** An n-gram is a contiguous sequence of *n* words.

- **Core Principle (Markov Assumption):** The probability of a word depends only on a fixed number (*n-1*) of preceding words.

$$P(w_k|w_1,...,w_{k-1}) \approx P(w_k|w_{k-n+1},...,w_{k-1})$$

  - Bigram (n=2): Probability depends on the previous word.
  - Trigram (n=3): Probability depends on the two previous words.

https://web.stanford.edu/~jurafsky/slp3/3.pdf9

Challenges of N-gram Models

- **Data Sparsity:** Many valid n-grams will not appear in the training corpus, leading to zero probability estimates.
    - **Solution: Smoothing:** Techniques like Laplace (add-one) and Kneser-Ney redistribute probability mass to unseen n-grams.
- **Short Context Window:** The fixed, short context ('n') prevents models from capturing long-range dependencies and deeper semantic meaning.
- **Result:** Generated text often lacked global coherence.

The Rise of Neural Networks

- **New Paradigm:** Deep learning promised to overcome the limitations of statistical models.

- **Key Advantages:**
  - Learned rich, distributed representations of language.
  - Captured dependencies over longer sequences.

- **Key Architectures:**
  - Word2vec, Glove, …
  - Recurrent Neural Networks (RNNs)
  - Long Short-Term Memory (LSTMs) & Gated Recurrent Units (GRUs)
  - Transformers

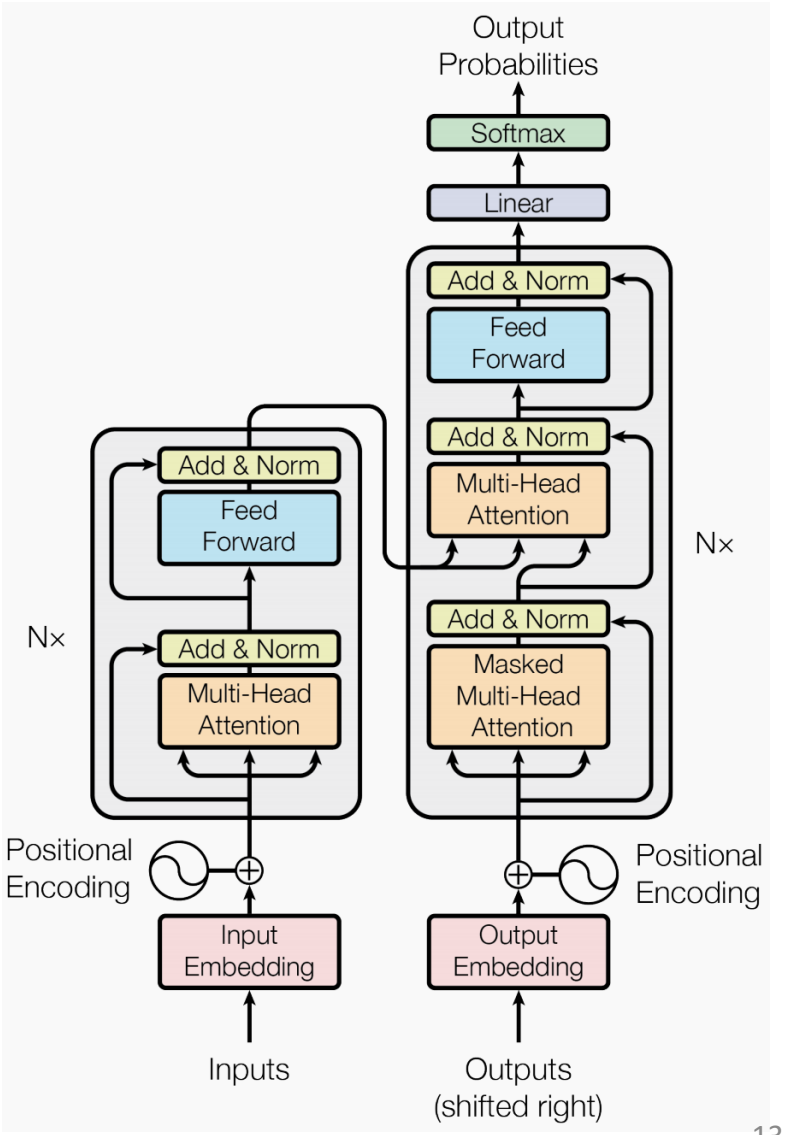
Summary of Eras

Feature/Era	Symbolic Era (1950s–1980s)	Statistical Era (1990s–2010s)	Recurrent Neural Era (2010s)	Transformer Era (2017–Present)
Core Principle	Hand-written linguistic & world knowledge rules	Markov assumption; learning statistical patterns from data	Gated memory cells to regulate information flow	Parallel self-attention mechanism
Key Models	ELIZA, SHRDLU	N-gram Models, Hidden Markov Models (HMMs)	Recurrent Neural Networks (RNNs), LSTMs, GRUs	Transformer, BERT, GPT, LLaMA
Strengths	High precision in narrow, predefined domains	Computationally simple, efficient for local patterns	Handles variable-length sequences, captures longer dependencies	Highly parallelizable, superior at long-range dependencies, scalable
Limitations	Brittle, not scalable, requires immense manual effort	Data sparsity, fixed short context, poor semantic understanding	Vanishing gradients (in simple RNNs), sequential computation bottleneck	Computationally expensive, requires massive data, lacks innate sequential bias

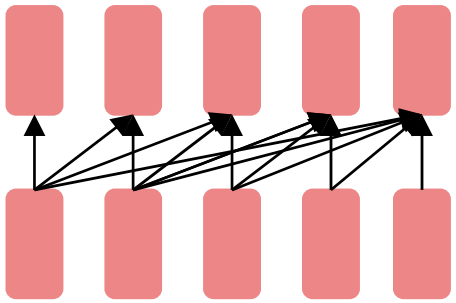
Transformer (2017- Present)

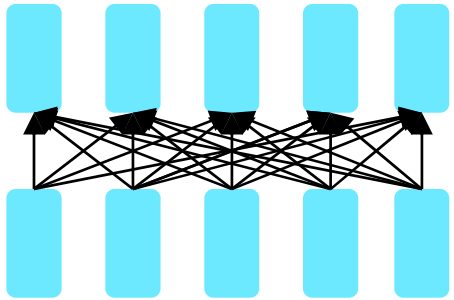
Attention Is All You Need

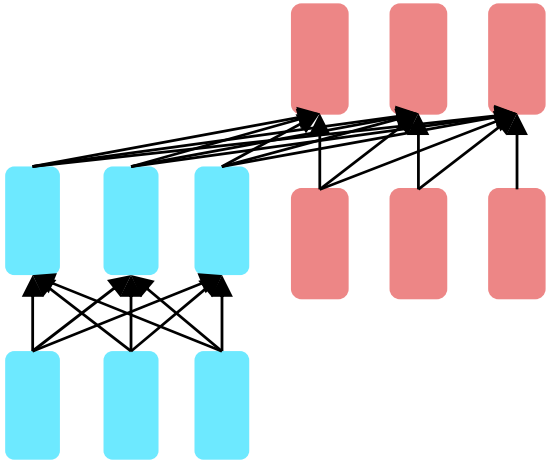
https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf



Three architectures for large language models







- **Decoders**

GPT, Claude, Mistral

Encoders

BERT family

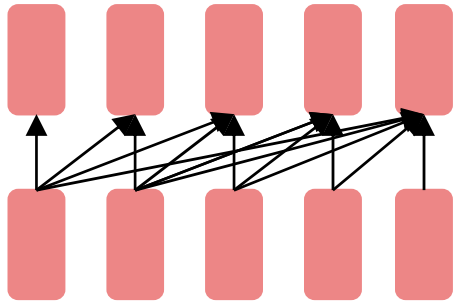
Encoder-decoders

Flan-T5, BART

Decoder-based Models

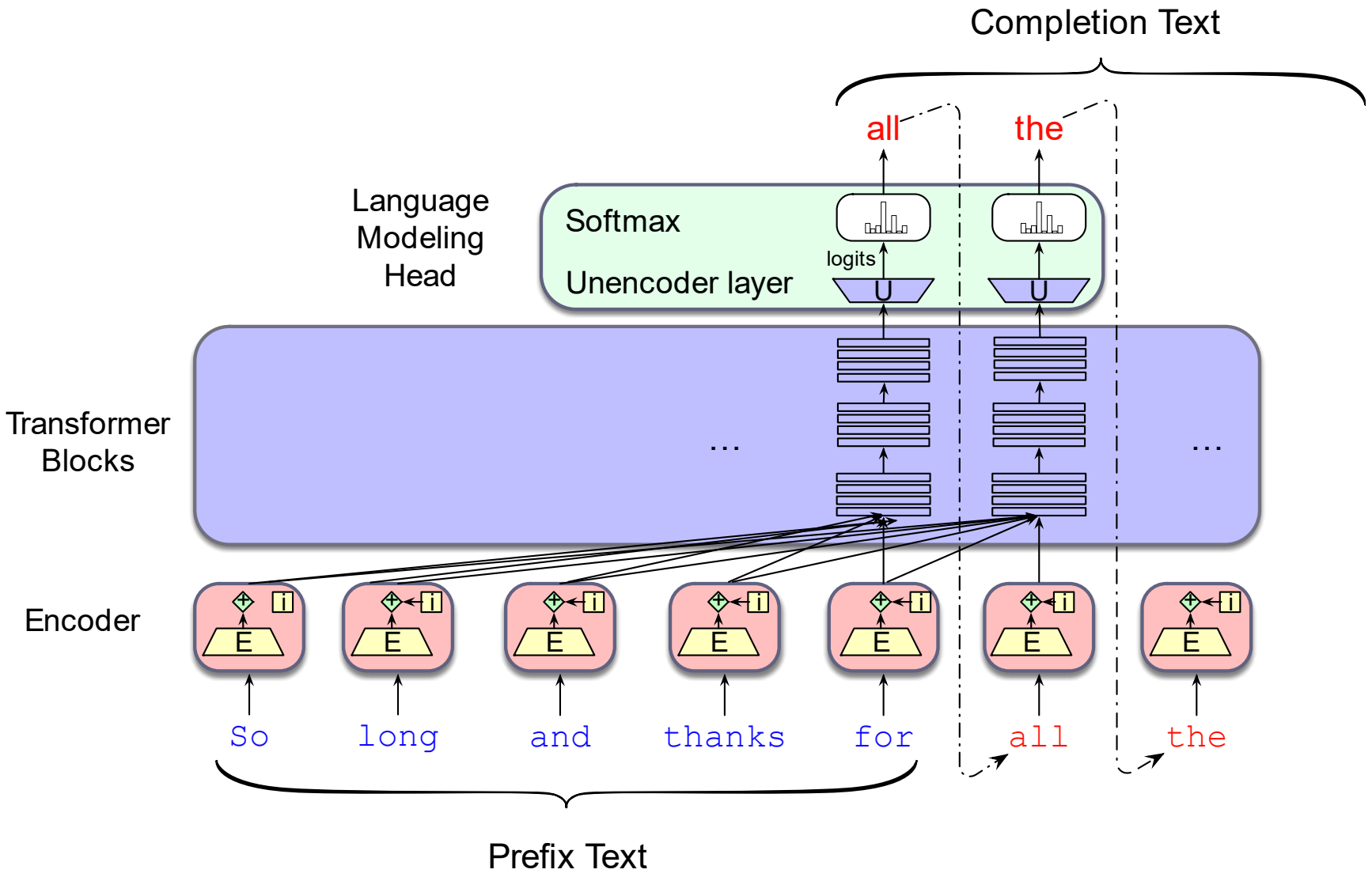
This lecture: decoder-only models

- Also called:
  - Causal LLMs
  - Autoregressive LLMs
  - Left-to-right LLMs



- **Predict words left to right**

Conditional Generation: Generating text conditioned on previous text!



Many practical NLP tasks can be cast as word prediction!

- Sentiment analysis: “**I like Jackie Chan**”

1. We give the language model this string:
   The sentiment of the sentence "I like Jackie Chan" is:

2. And see what word it thinks comes next:

$P(\text{positive} \mid \texttt{The sentiment of the sentence ``I like Jackie Chan" is:})$

$P(\text{negative} \mid \texttt{The sentiment of the sentence ``I like Jackie Chan" is:})$

Framing lots of tasks as conditional generation

- **QA: “Who wrote The Origin of Species”**

1. We give the language model this string:

   `Q: Who wrote the book ``The Origin of Species"? A:`

2. And see what word it thinks comes next:

   $P(w|$`Q: Who wrote the book ``The Origin of Species"? A:`$)$

3. And iterate:

   $P(w|$`Q: Who wrote the book ``The Origin of Species"? A: Charles`$)$

Summarization

Original Article

The only thing crazier than a guy in snowbound Massachusetts boxing up the powdery white stuff and offering it for sale online? People are actually buying it. For $89, self-styled entrepreneur Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says.

But not if you live in New England or surrounding states. "We will not ship snow to any states in the northeast!" says Waring's website, ShipSnowYo.com. "We're in the business of expunging snow!"

His website and social media accounts claim to have filled more than 133 orders for snow – more than 30 on Tuesday alone, his busiest day yet. With more than 45 total inches, Boston has set a record this winter for the snowiest month in its history. Most residents see the huge piles of snow choking their yards and sidewalks as a nuisance, but Waring saw an opportunity.

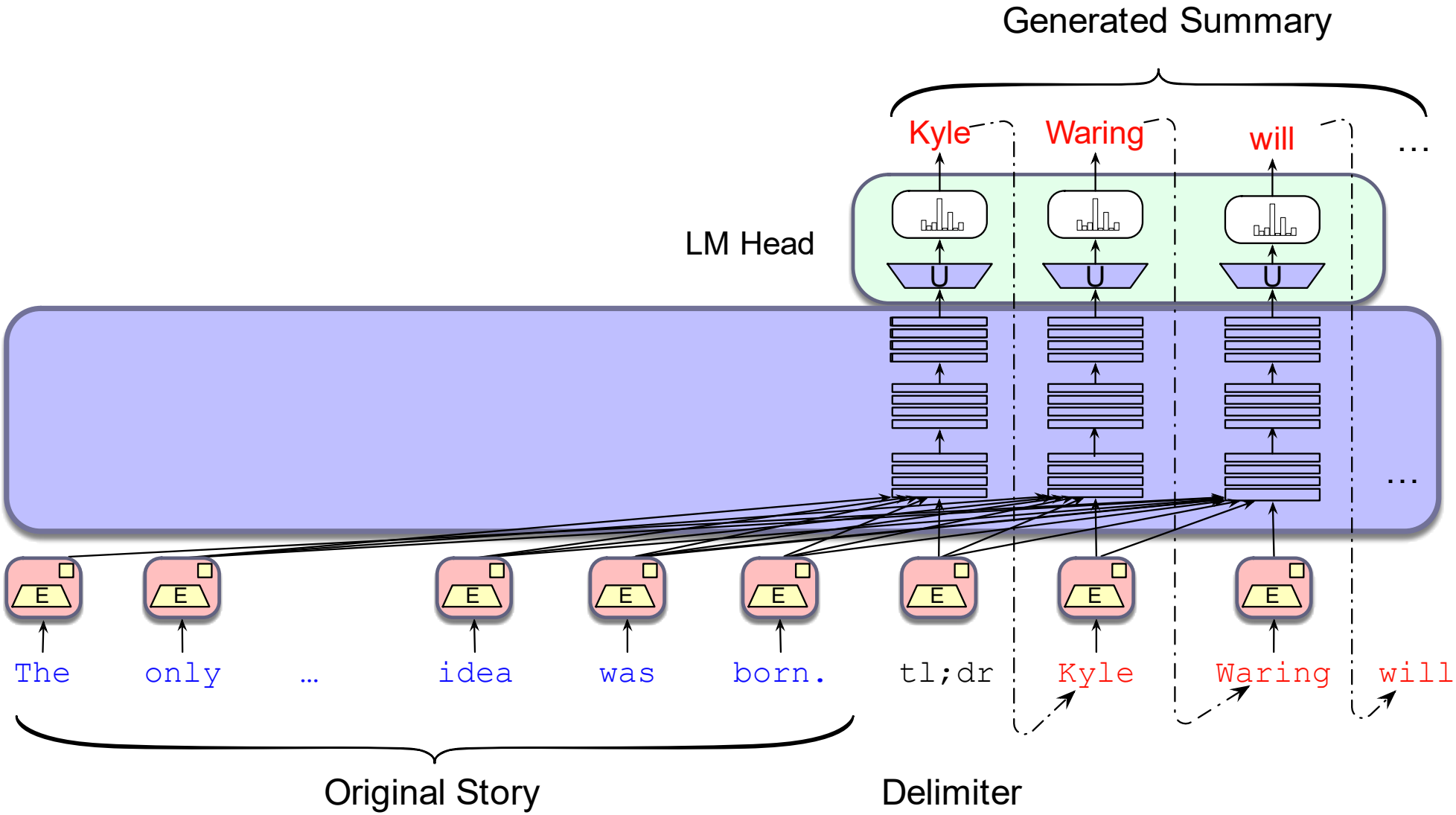
According to Boston.com, it all started a few weeks ago, when Waring and his wife were shoveling deep snow from their yard in Manchester-by-the-Sea, a coastal suburb north of Boston. He joked about shipping the stuff to friends and family in warmer states, and an idea was born. [...]

Summary

Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says. But not if you live in New England or surrounding states.

Figure 10.2 Excerpt from a sample article and its summary from the CNN/Daily Mail summarization corpus (Hermann et al., 2015), (Nallapati et al., 2016).

LLMs for summarization (using tl;dr)



Sampling for LLM Generation

Decoding and Sampling

- This task of choosing a word to generate based on the model's probabilities is called **decoding**.

- The most common method for decoding in LLMs: **sampling**.

- Sampling from a model's distribution over words means:
  - to choose random words according to their probability assigned by the model.

- After each token we'll sample words to generate according to their probability ***conditioned*** *on our previous choices*,
  - A transformer language model will give the probability

Random sampling

$$
\begin{aligned}
&\mathrm{i} \leftarrow 1 \\
&w_i \sim \mathrm{p}(w) \\
&\textbf{while } w_i \text{ != EOS} \\
&\quad \mathrm{i} \leftarrow \mathrm{i} + 1 \\
&\quad w_i \sim \mathrm{p}(w_i \mid w_{<i})
\end{aligned}
$$

Random sampling doesn't work very well

- Even though random sampling mostly generate sensible, high-probable words,
- There are many odd, low- probability words in the tail of the distribution
- Each one is low- probability but added up they constitute a large portion of the distribution
- So they get picked enough to generate weird sentences

Factors in word sampling: *quality* and *diversity*

- Emphasize **high-probability** words
  - + **quality**: more accurate, coherent, and factual,
  - **- diversity**: boring, repetitive.

- Emphasize **middle-probability** words
  - + **diversity**: more creative, diverse,
  - - **quality**: less factual, incoherent

Top-k sampling:

Top-k sampling is a simple generalization of greedy decoding

1.Choose # of words *k*

2. For each word in the vocabulary V , use the language model to compute the likelihood of this word given the context $p(w_t \mid w_{<t})$

3. Sort the words by likelihood, keep only the top *k* most probable words.

4. Renormalize the scores of the *k* words to be a legitimate probability distribution.

5. Randomly sample a word from within these remaining *k* most-probable words according to its probability.

When k = 1, top-k sampling is identical to greedy decoding.

Top-p sampling (= nucleus sampling)

Holtzman et al., 2020

- **Problem with top-*k*:** ***k*** is fixed so may cover very different amounts of probability mass in different situations
- **Idea:** Instead, keep the top **p** percent of the probability mass

- Given a distribution $P(w_t \mid \mathbf{w}_{<t})$, the top-$p$ vocabulary $V^{(p)}$ is the smallest set of words such that

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p$$

Top-p sampling (= nucleus sampling)

Holtzman et al., 2020

Suppose the model predicts these probabilities for words in a given context:

$$p(w) = \{\text{"cat"}: 0.5, \text{"dog"}: 0.3, \text{"bird"}: 0.15, \text{"fish"}: 0.05\}.$$

Step 1: Using Top-k Sampling (e.g., $k = 2$):

- Top-k keeps only the top 2 words by probability:
$$\text{"cat"}: 0.5, \text{"dog"}: 0.3.$$
- The rest ("bird" and "fish") are ignored, even though together they have a 20% chance.

Step 2: Using Top-p Sampling (e.g., $p = 0.9$):

- Top-p keeps words until their cumulative probability reaches 0.9:
  - "cat" = 0.5 (50%)
  - "dog" = 0.3 (30%), cumulative = 0.8
  - "bird" = 0.15 (15%), cumulative = **0.95** (exceeds 0.9, stop here)
- The result:
$$\text{"cat"}: 0.5, \text{"dog"}: 0.3, \text{"bird"}: 0.15.$$
- Only "fish" (0.05) is excluded because it contributes too little.

•**Dynamic Selection**: Top-p adapts to the context:
• if probabilities are concentrated on a few words, it selects fewer.
• If probabilities are spread out, it includes more.

Flexibility: Unlike Top-k, it avoids cutting off useful low-probability words when the distribution is flatter

Temperature sampling

- Reshape the distribution instead of truncating it

- Divide the logit by a temperature parameter τ before passing it through the softmax.

- Instead of $\mathbf{y} = \mathrm{softmax}(u)$

- We do $\mathbf{y} = \mathrm{softmax}(u/\tau)$

Temperature sampling

Intuition from Thermodynamics:

- **Low Temperature**: ($\tau \leq 1$)
  - A cooler system explores only **low-energy (better)** states (less randomness).
  - In language generation, this focuses on coherence but sacrifices diversity.
- **High Temperature ($\tau > 1$)**:
  - Think of a system at high heat—it explores **many possible states** (more randomness).
  - In language generation, this creates more creative but potentially less coherent text.

Temperature sampling

$$\mathbf{y} = \mathbf{softmax}(\boldsymbol{u}/\boldsymbol{\tau}) \qquad 0 \leq \tau \leq 1$$

- **Why does this work?**
- When τ is close to 1 the distribution doesn't change much.
- The lower τ is, the larger the scores being passed to the softmax
- Softmax pushes high values toward 1 and low values toward 0.
- Large inputs pushes high-probability words higher and low probability word lower, making the distribution more greedy.
- As τ approaches 0, the probability of most likely word approaches 1

Temperature sampling: Example

- https://drive.google.com/file/d/1nP85AK10boNLMn58euA-M2jTxp_SfbwU/view?usp=sharing

LLMs Training

Large Language Models - Training

1. Pretraining using Self-supervised learning
2. Fine-tuning
3. Reinforcement learning from human feedback (Alignment with human values)
   - nudging the LLM towards values you desire

Pretraining

- The big idea that underlies all the amazing performance of language models

- First **pretrain** a transformer model on enormous amounts of text
- Then **apply** it to new tasks.

Self-supervised training algorithm

- We just train them to predict the next word!

1. Take a corpus of text
2. At each time step *t*
   i. ask the model to predict the next word
   ii. train the model to minimize the error in this prediction

"Self-supervised" because it just uses the next word as the label!

Intuition of language model training: loss

- Same loss function: **cross-entropy loss**
  - We want the model to assign a high probability to true word *w*
  - = want loss to be high if the model assigns too low a probability to w

- **CE Loss:** The negative log probability that the model assigns to the true next word w
  - If the model assigns too low a probability to w
  - We move the model weights in the direction that assigns a higher probability to w

Cross-entropy loss for language modeling

- **CE loss**: difference between the correct probability distribution and the predicted distribution

$$L_{CE} = -\sum_{w \in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w]$$

- The correct distribution $\mathbf{y}_t$ knows the next word, so is 1 for the actual next word and 0 for the others.
- So in this sum, all terms get multiplied by zero except one: the logp the model assigns to the correct next word, so:

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

Teacher forcing

- At each token position *t*, model sees correct tokens $w_{1:t}$,
  - Computes loss (–log probability) for the next token w_{t+1}

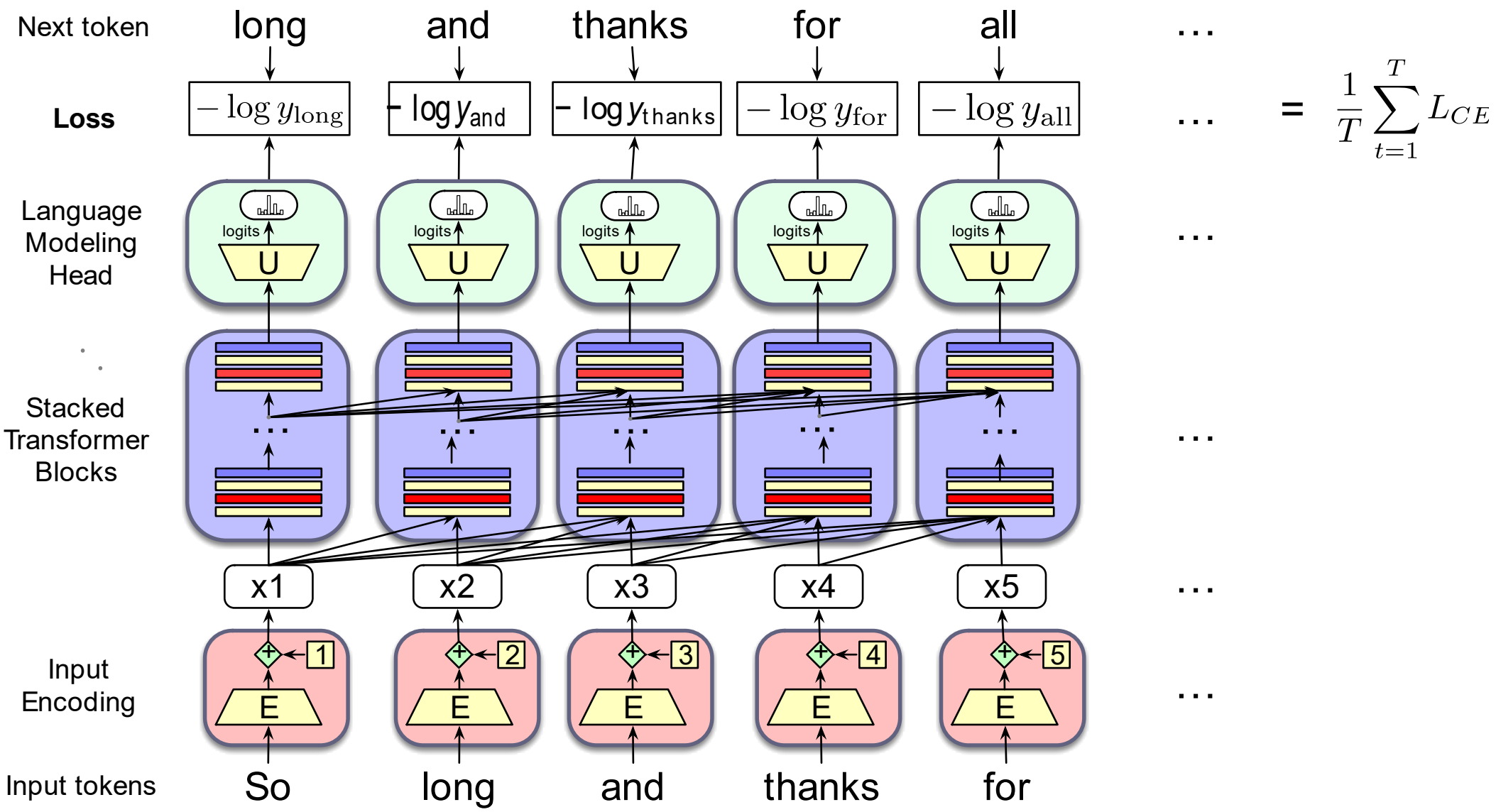
- At next token position t+1 we ignore what model predicted for w_{t+1}
  - Instead we take the **correct** word w_{t+1}, add it to context, move on

Exposure Bias Problem

Fine-tuning a large language model with reinforcement learning for educational question generation

S Lamsiyah, A El Mahdaouy, A Nourbakhsh, C Schommer International Conference on Artificial Intelligence in Education, 424-438

Training a transformer language model

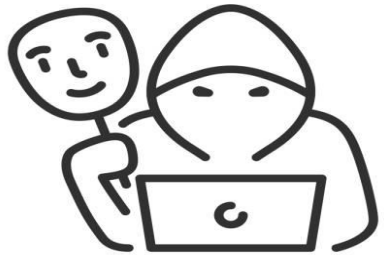


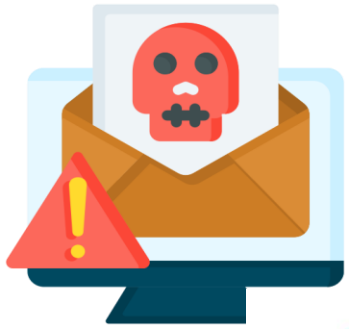
Sources of LLMs'Strong Generation Capabilities

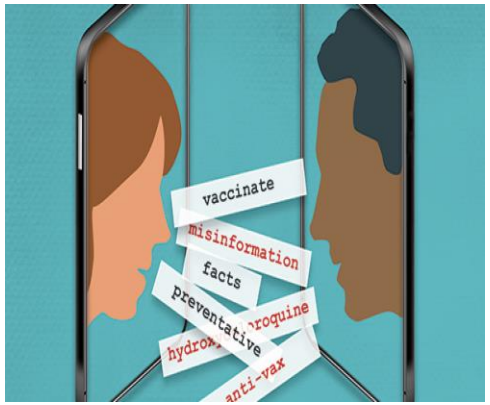
- **Pre-training on massive Corpora**
- **Fine-tuning/ Instruction Tuning**
- **In-Context Learning (ICL)**
- **Alignment with Human feedback**
- **Complex reasoning capabilities**
- **….**

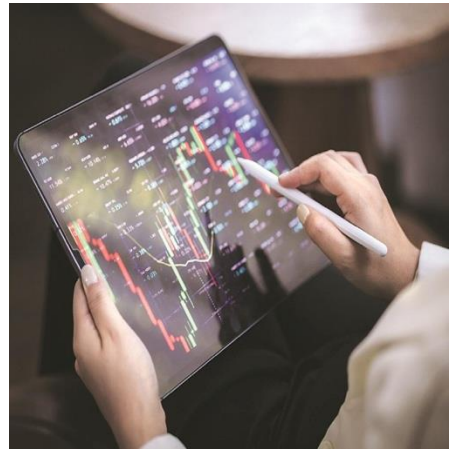
Part 2: AI-Generated Text Detection Methods

Why AI-Generated Text Detection Matters?



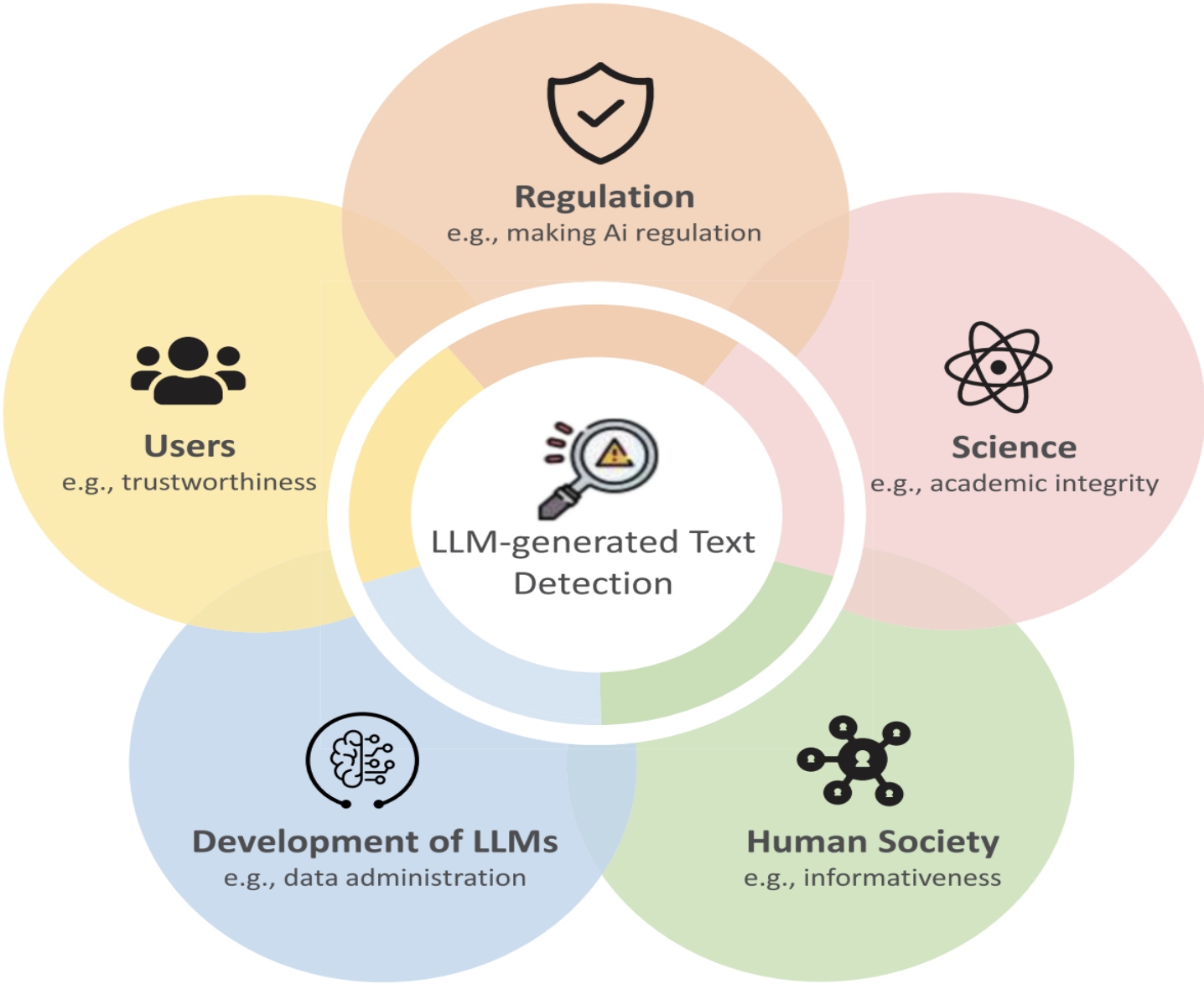






Why AI-Generated Text Detection Matters?

https://aclanthology.org/2025.cl-1.8.pdf



Problem Formulation

Problem Definition: AI-Generated Text Detection (AGTD)

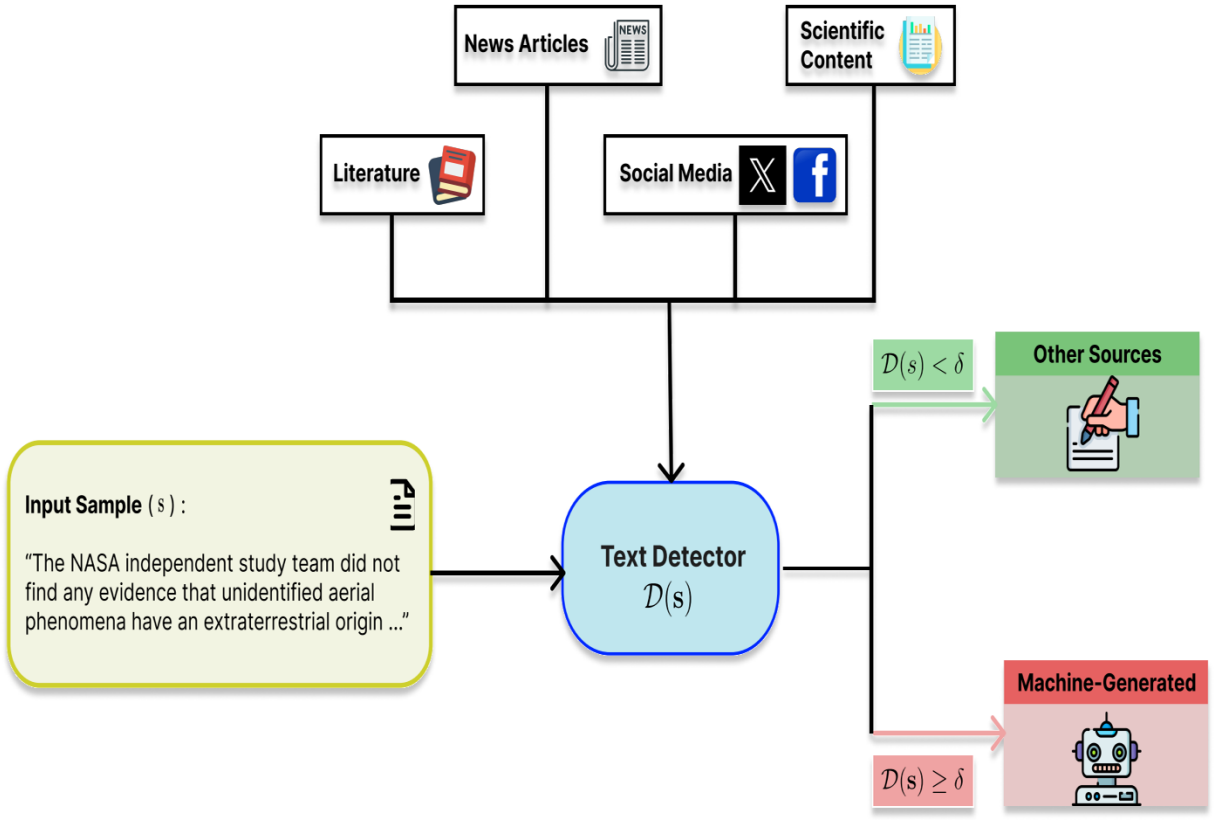
- ***Task Type:*** *Binary Classification* *(*Determine if a passage is **human-written** or **AI-generated**)

- **Formal Setup:**
  - Given input text: $\boldsymbol{s}$
  - Detector function: $\boldsymbol{D(s)} \in \mathbb{R}$ (outputs a score)
  - Decision rule:

    $\mathbf{D(s)} \geq \boldsymbol{\delta}$ $\Rightarrow$ **AI-generated**

    $\mathbf{D(s)} < \boldsymbol{\delta}$ $\Rightarrow$**Human-written**

- **Key Idea:** Compare detector score against a threshold $\boldsymbol{\delta}$ to classify text origin.



https://proceedings.mlr.press/v202/mitchell23a.htm

Problem Definition: AI-Generated Text Detection (AGTD)

- **Multi-class detection**
  - classify *which AI model family* generated the text (e.g., GPT vs LLaMA vs Mistral).
  - Useful for forensic or attribution tasks.

- **Open-world detection**
  - Models must detect AI-generated text from **unseen generators or domains** (not included in training).
  - Example: Detector trained on GPT-3 but tested on GPT-4 or domain-shifted text.

- **Mixed-text detection**
  - Some texts are partly human, partly AI (e.g., student writes intro, GPT writes conclusion).
  - Task: **find the change point** (where the switch occurs).

AI-Generated Text Detection Methods taxonomy

AGTD Methods

Watermarking-based Approach

- **Model-Driven Watermarking**
- **Data-Driven Watermarking**
- **Post-Processing Watermarking**

Statistical-based Approach

- **Linguistic and Stylometric Feature**
- **White-box Statistics**
- **Black-box statistics**

Neural-based Approach

- **Feature-based Classification**
- **Pre-training and Fine-tuning**
- **LLMs as detectors**

Watermarking-based Approach

- Embeds hidden, machine-detectable signals at text creation.
- Proactive: adds a traceable signature, unlike passive detection.
- Enables reliable verification of origin and authorship.
- Core Challenge: balancing **detectability, text quality, and robustness**

Research on text watermark algorithm for generative large language models
https://lutpub.lut.fi/handle/10024/169392?show=full

Watermarking-based Approach
Foundational Principles

- A watermarking system has **two parts**:

- **Generator** $(\boldsymbol{A})$ – the LLM is nudged to embed a hidden signal while generating text.
  - Input = prompt x + watermark message w
  - Output = watermarked text t
  - Formula: $A(x, w) = t$

- **Detector** $(\boldsymbol{D})$ – later, another system checks the text for the hidden signal.
  - Input = text t
  - Output = predicted watermark message $\hat{w}$ (or *null* if none found).
  - Formula: $D(t) = \hat{w}$

Watermarking-based Approach
Foundational Principles

- **Generator A:** the LLM at inference time, embedding the watermark while generating text.

- **Detector D:** a statistical test (or classifier) that can later confirm the watermark in the text

Watermarking-based Approach
Examples

Step 1 - Generation (Watermark Generator A):

- ***Prompt x:*** *"Write me a short story about a robot and a cat."*
- Watermark message w: *"OpenAI signature."* (hidden)
- LLM outputs a normal-looking story: *"The robot wandered into the alley where a curious cat followed. Together, they discovered…"*
- Behind the scenes → the LLM has been biased to choose certain "green" words slightly more often.
- To the human, it looks like a normal story.

Step 2 - Detection (Watermark Detector D):

- Detector D analyzes the word sequence.
- It recomputes the hidden "green/red" partitions and checks if the story has too many "green" words.
- If yes → it outputs $\widehat{w}$ = *"OpenAI signature."*
- If no → it outputs *null* (text is unmarked).

Watermarking-based Approach
Key Characteristics

1. **Imperceptibility (Low Impact on Quality)** :

- The watermarking process should not perceptibly degrade the quality of the text.

$$\forall w_i, R(A(x, \emptyset), A(x, w_i)) < \delta$$

- Where R is a quality evaluation function and δ is a small threshold.

A Survey of Text Watermarking in the Era of Large Language Models
https://arxiv.org/pdf/2312.07913

Watermarking-based Approach
Key Characteristics

2. Robustness:

- Watermark must remain detectable after transformations/attacks:
    - Compression, Noise injection, Cropping, **Paraphrasing** (most critical for text)

- Measured by the ***Bit Error Rate (BER)***:

$$BER = \frac{1}{N}\sum_{i=1}^{N} 1(m_i \neq \hat{m}_i)$$

- m = the original **watermark message** embedded in the text
- $\hat{m}_i$ = the **extracted message** that the detector recovers after possible attacks
- N = the total number of bits (pieces of information) in the watermark message
- $1(m_i \neq \hat{m}_i)$ = an indicator function:
    - It equals **1** if the recovered bit is different from the original.
    - It equals **0** if they match.

A Survey of Text Watermarking in the Era of Large Language Models https://arxiv.org/pdf/2312.07913

Watermarking-based Approach

Key Characteristics

- Suppose the watermark message m is:

$$m = [1,0,1,1,0]$$

- And after paraphrasing, the detector extracts:

$$\hat{m} = [1,1,1,0,0]$$

- Compare each bit:
  - bit 1: correct ✅
  - bit 2: wrong ❌
  - bit 3: correct ✅
  - bit 4: wrong ❌
  - bit 5: correct ✅

So, 2 out of 5 bits are wrong.
BER = 2/5= 0.4 => 40% error rate

Lower BER → higher robustness

Watermarking-based Approach
Key Characteristics

3. **Security =** The watermark must be **resistant to removal or forgery** by adversaries and attacks (e.g. Synonym substitution, GAN-based watermark removal, Adversarial paraphrasing)

- **Evaluation Metrics:**
  - **Attack Success Rate (ASR):** % of adversarially modified texts where watermark is undetectable

Watermarking for AI Content Detection: A Review on Text, Visual, and Audio Modalities
https://arxiv.org/pdf/2504.03765v1

Watermarking-based Approach
Key Characteristics

4. **Capacity** = ability to embed **sufficient payload information** for downstream use:
   - Source tracking
   - User identification
   - Multi-party attribution

- Must **not violate imperceptibility constraint** (text must remain natural)
- **Evaluation Metrics:**
  - **Payload size (bits per token / bits per word)**
  - **Embedding rate:** ratio of embedded information to total text length
- **Capacity–Quality Trade-off:** correlation between payload size and perplexity/fluency degradation

https://arxiv.org/pdf/2504.03765v1

Watermarking-based Methods

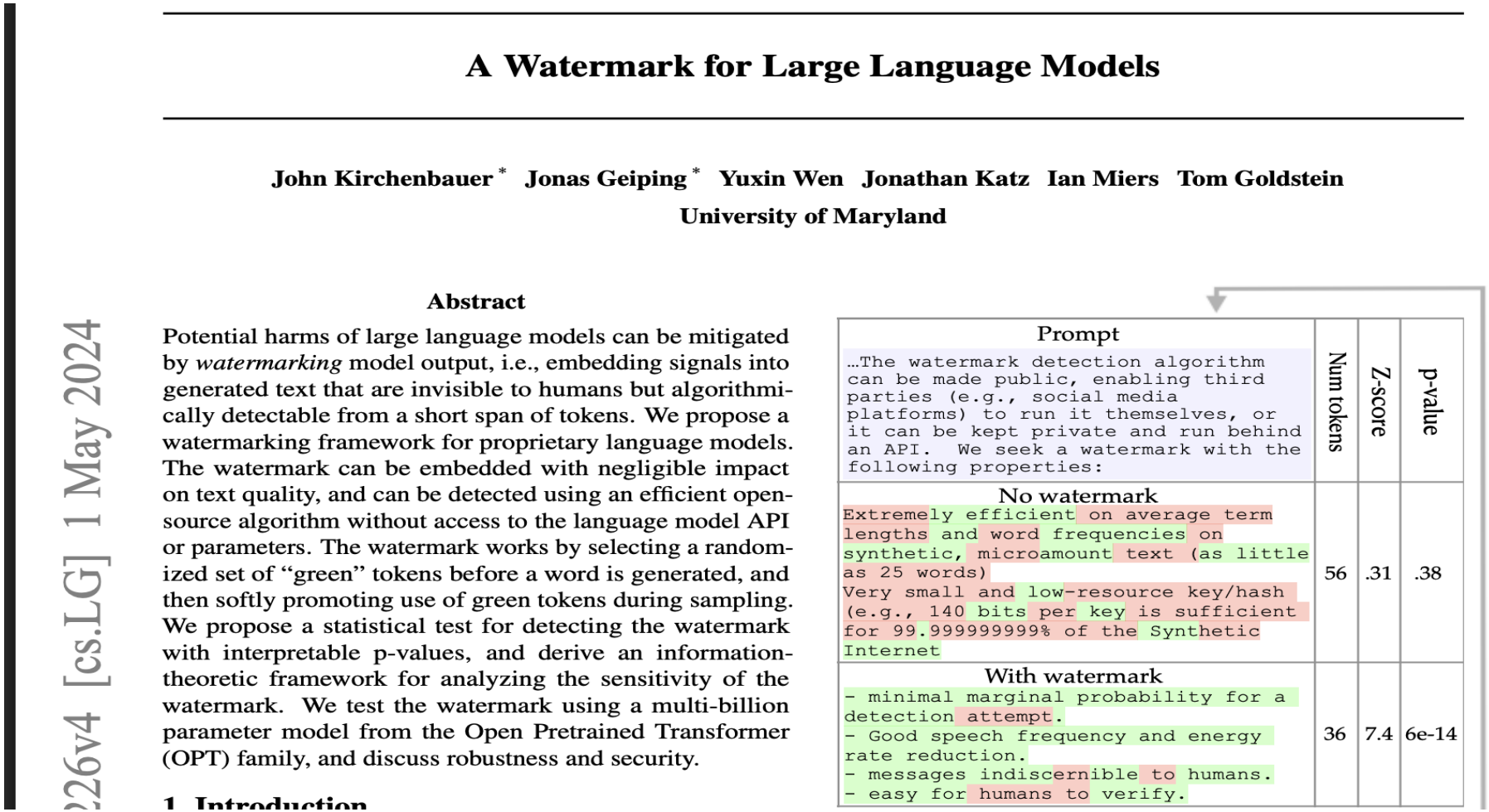
1. **Model-Driven Watermarking**
2. **Data-Driven Watermarking**
3. **Post-Processing Watermarking**

A Survey of Text Watermarking in the Era of Large Language Models
https://arxiv.org/pdf/2312.07913

Model-Driven Watermarking

- Model-driven (generative) watermarking integrates watermarking into decoding stage
- Requires white-box access to model logits
- Alters token probabilities during generation
- Seminal method: ***Red-Green list algorithm (Kirchenbauer et al., 2023)***

Model-Driven Watermarking



26v4 [cs.LG] 1 May 2024

A Watermark for Large Language Models

John Kirchenbauer * Jonas Geiping * Yuxin Wen Jonathan Katz Ian Miers Tom Goldstein
University of Maryland

Abstract

Potential harms of large language models can be mitigated by *watermarking* model output, i.e., embedding signals into generated text that are invisible to humans but algorithmically detectable from a short span of tokens. We propose a watermarking framework for proprietary language models. The watermark can be embedded with negligible impact on text quality, and can be detected using an efficient open-source algorithm without access to the language model API or parameters. The watermark works by selecting a randomized set of "green" tokens before a word is generated, and then softly promoting use of green tokens during sampling. We propose a statistical test for detecting the watermark with interpretable p-values, and derive an information-theoretic framework for analyzing the sensitivity of the watermark. We test the watermark using a multi-billion parameter model from the Open Pretrained Transformer (OPT) family, and discuss robustness and security.

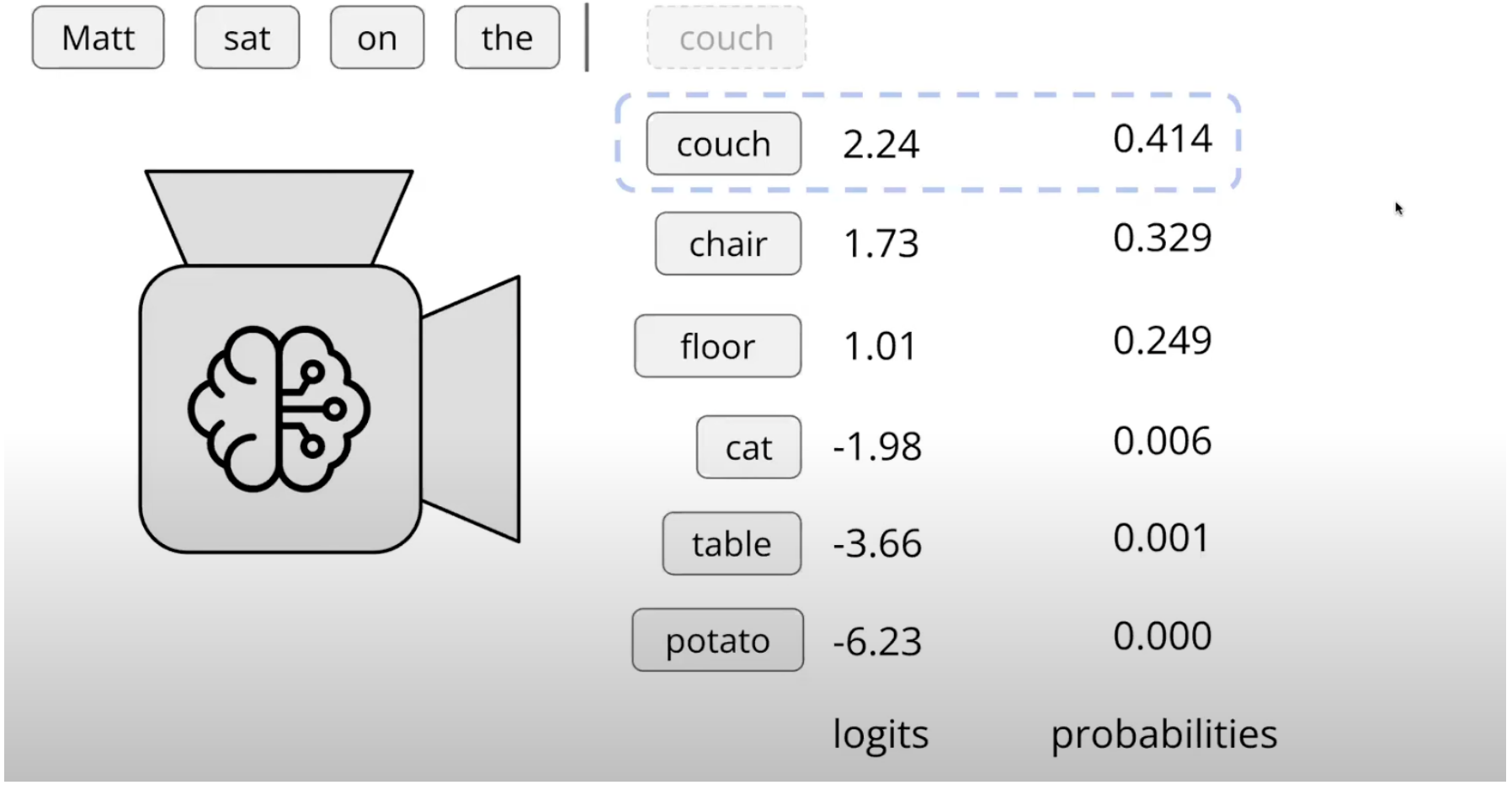
1 Introduction

Prompt	Num tokens	Z-score	p-value
…The watermark detection algorithm can be made public, enabling third parties (e.g., social media platforms) to run it themselves, or it can be kept private and run behind an API. We seek a watermark with the following properties:			
No watermark Extremely efficient on average term lengths and word frequencies on synthetic, microamount text (as little as 25 words) Very small and low-resource key/hash (e.g., 140 bits per key is sufficient for 99.999999999% of the Synthetic Internet	56	.31	.38
With watermark - minimal marginal probability for a detection attempt. - Good speech frequency and energy rate reduction. - messages indiscernible to humans. - easy for humans to verify.	36	7.4	6e-14

A Watermark for Large Language Models
https://proceedings.mlr.press/v202/kirchenbauer23a.html

Model-Driven Watermarking

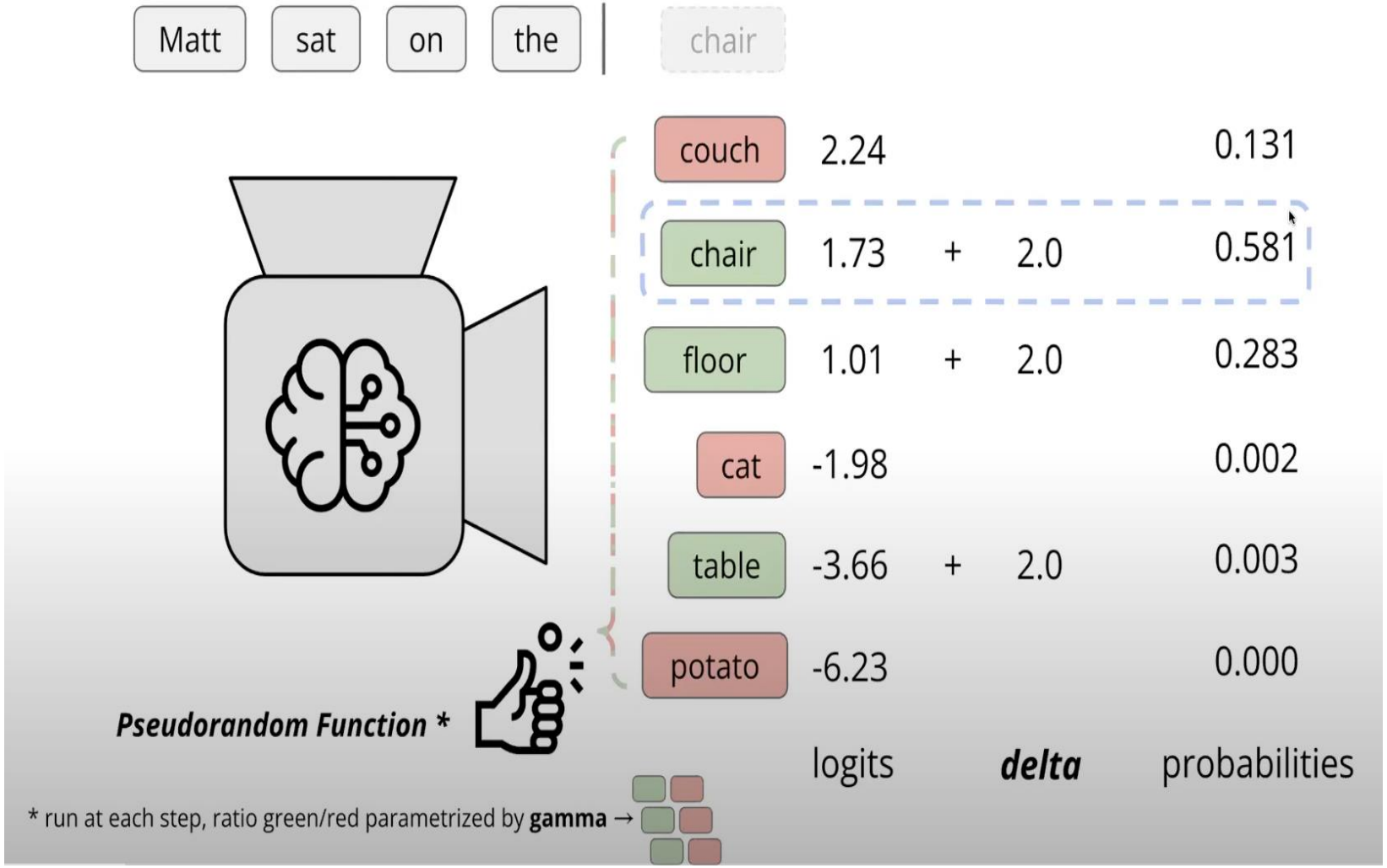
Next Token Prediction Language Modeling



A Watermark for Large Language Models
https://proceedings.mlr.press/v202/kirchenbauer23a.html

Model-Driven Watermarking: **Red-green list**

Embedding the watermark via softly biasing the logits



A Watermark for Large Language Models
https://proceedings.mlr.press/v202/kirchenbauer23a.html

Model-Driven Watermarking:
Red-green list Algorithm

1. List Generation

- At step t, vocabulary V split into into a **"green list" (V_G)** and a **"red list" (V_r)**
  - Partitioning uses a cryptographic hash function of previous token (x_{t-1})
  - Green list defined as:

$$V_G = \{ v \in V \mid h(x_{t-1}), v) \leq \gamma \cdot H_{max} \}$$

  - γ controls proportion of tokens in $\boldsymbol{V_G}$
- Red list is complement: $V_R = V \setminus V_G$

Model-Driven Watermarking: Red-green list Algorithm

2. Logit Modification:

- Once the lists are generated, the logits produced by the LLM are modified to favor tokens from the green list. This can be done in two primary ways:

- ***Soft Watermark:***

$$l_{i'} = li + \delta \; if \; vi \in VG, else \; li$$
$$p' = softmax(l')$$

- Increases likelihood of V_G tokens while preserving quality

- ***Hard Watermark:***

$$l_i' = -\infty \; if \; vi \in VR \quad li,$$
$$= li \; if \; vi \in VG$$

- Stronger signal but can degrade fluency

Model-Driven Watermarking:
Red-green list Algorithm

3. Detection Mechanism

- Recompute V_G / VR at each step using hash
- Count tokens in green list: N_G
- Null hypothesis: $N_G \sim Binomial(T, \gamma)$
- One-sided z-test:

$$z = \frac{(N_G - T_\gamma)}{\sqrt{(T_{\gamma(1-\gamma)})}}$$

- T = the number of tokens in the text
- G = Green tokens in the text

- High z-score (e.g., z > 4) ⇒ strong evidence of watermark

Model-Driven Watermarking:
Red-green list Algorithm

- No retraining – only sampling modified
- Minimal impact on text quality
- Efficient detection using only text
- Statistically interpretable with p-values

- TL;DR Algorithm
  - Generation: Partition vocab → boost Green tokens → sample
  - Detection: Recompute partitions → count Green tokens → z-test

Watermarking-based Methods

1. Model-Driven Watermarking

2. **Data-Driven Watermarking**

A Survey of Text Watermarking in the Era of Large Language Models
https://arxiv.org/pdf/2312.07913

Data-Driven Watermarking

- Paradigm shift: signal embedding through data, not decoding
- Watermarks linked to semantics/content
- Achieved during training or via inference-time prompts
- **Goal**: more robust, semantically grounded watermarks

Data-Driven Watermarking

1. **Training-Based:** Involves "poisoning" the pre-training data with specific, detectable information, such as plausible but fictitious knowledge.

2. **Dynamic Prompt-Based:** Uses a secondary LLM to generate specific instructions that guide the primary LLM to produce text with subtle, pre-defined patterns.

3. **Semantic Watermarking (SemaMark):** Partitions the vocabulary based on the semantic similarity of tokens to the preceding context, making the watermark more robust to paraphrasing attacks.

Watermarking Language Models through Language Models
https://arxiv.org/pdf/2411.05091v1

Data-Driven Watermarking
Training-Based Signature Insertion

- Insert watermarks by 'poisoning' pre-training data
- Add plausible but fictitious knowledge (e.g., fake historical figure)
- If LLM recalls this knowledge → watermark detected
- Effectiveness ↑ with density & diversity of injected passages

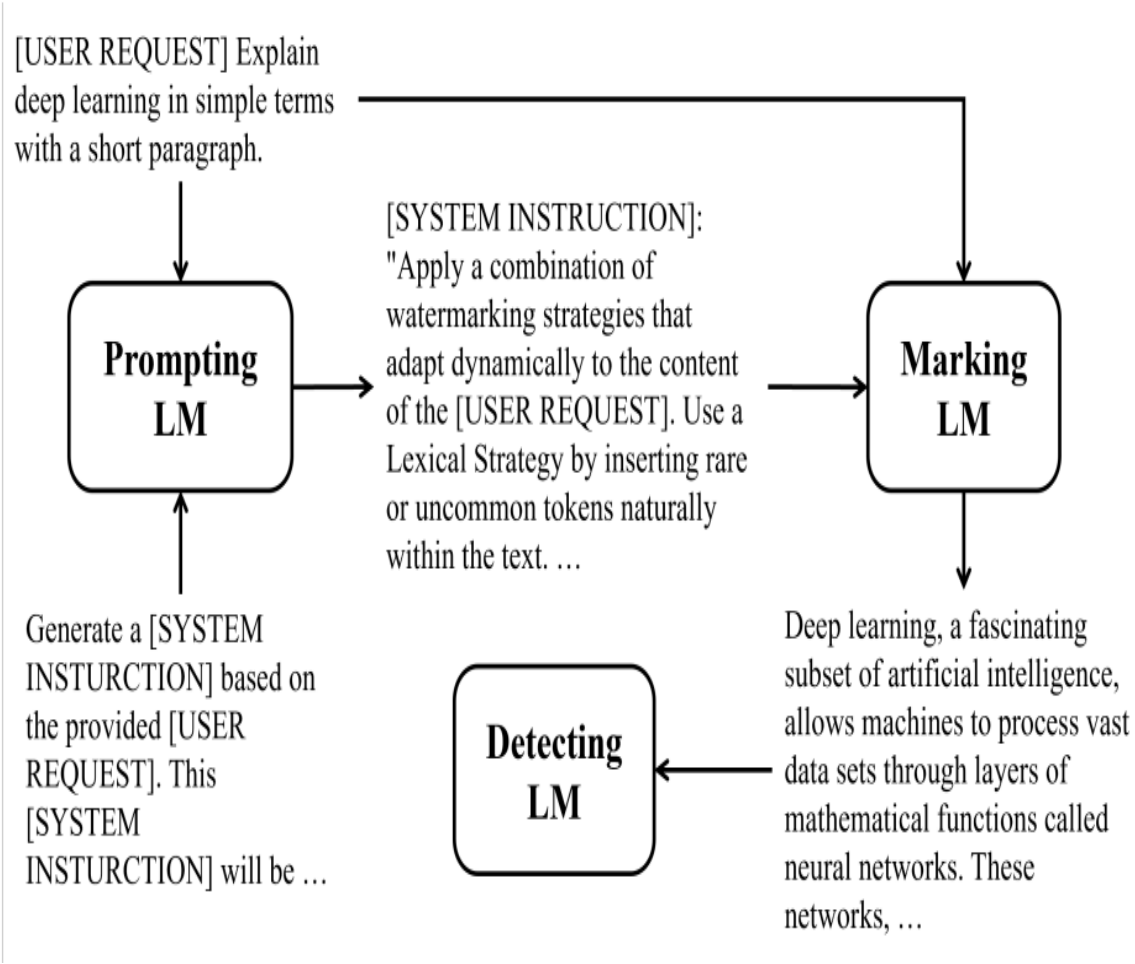
Example: Robust Data Watermarking in Language Models by Injecting Fictitious Knowledge https://aclanthology.org/2025.findings-acl.736/

Data-Driven Watermarking
Dynamic Prompt-Based Watermarking

- Uses inference-time prompts, no retraining required

- Pipeline:
  - Prompting LM generates a system instruction
  - Marking LM outputs response with subtle watermark patterns
  - Detecting LM trained to classify watermarked vs. non-watermarked

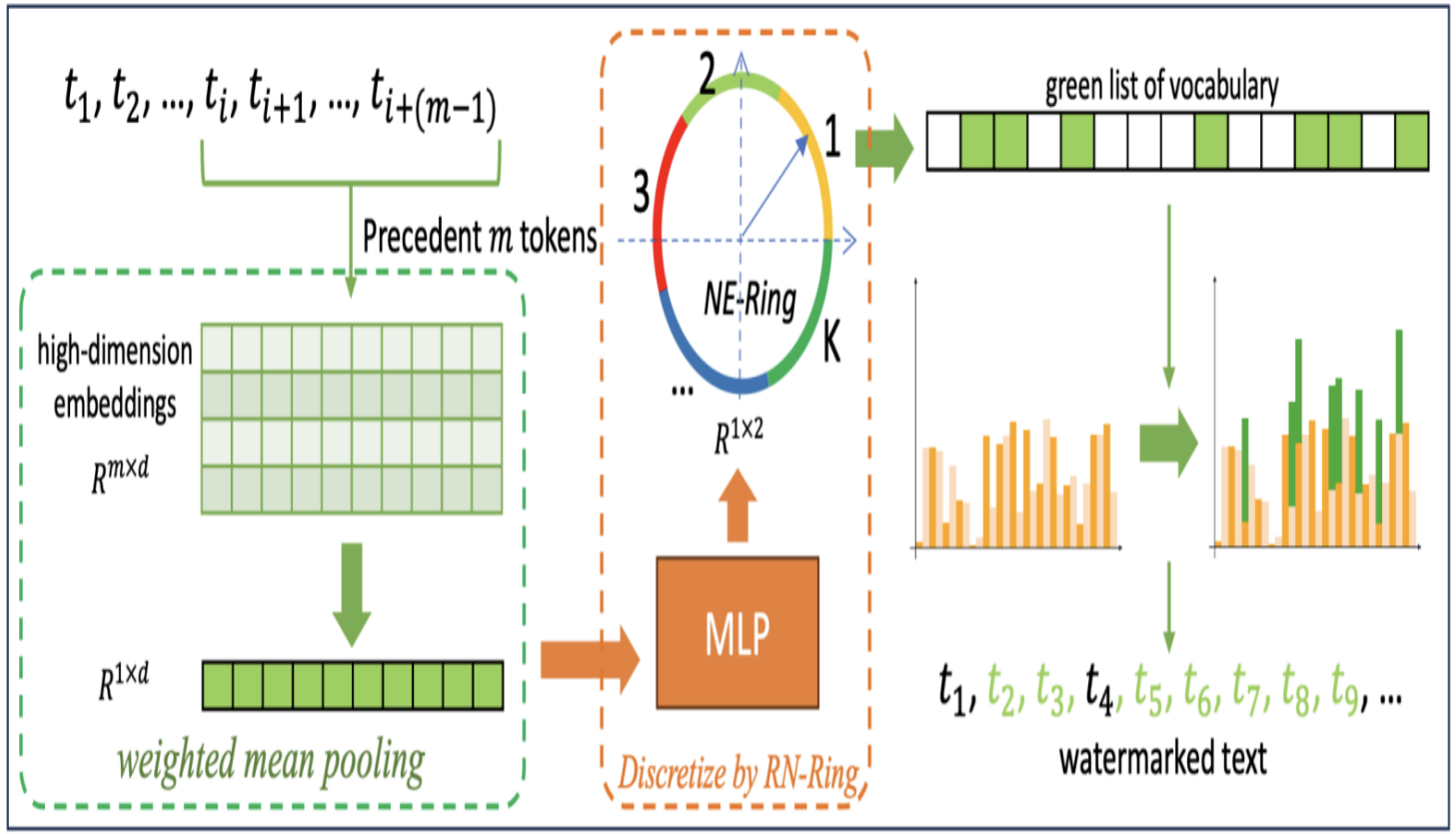
- Relies on curated dataset for detector training



Watermarking Language Models through Language Models
https://arxiv.org/pdf/2411.05091v1

Data-Driven Watermarking
Semantic Watermarking SemaMark'2024

- Addresses weakness of hash-based methods against paraphrasing
- Uses semantic embeddings of context to partition vocab
- Candidate tokens grouped by semantic similarity
- Meaning preserved even under paraphrasing → watermark survives
- More robust to semantic-preserving attacks



A Robust Semantics-based Watermark for Large Language Models against Paraphrasing
https://aclanthology.org/2024.findings-naacl.40.pdf

Watermarking-based Methods

1. Model-Driven Watermarking
2. Data-Driven Watermarking
3. **Post-Processing Watermarking**

A Survey of Text Watermarking in the Era of Large Language Models
https://arxiv.org/pdf/2312.07913

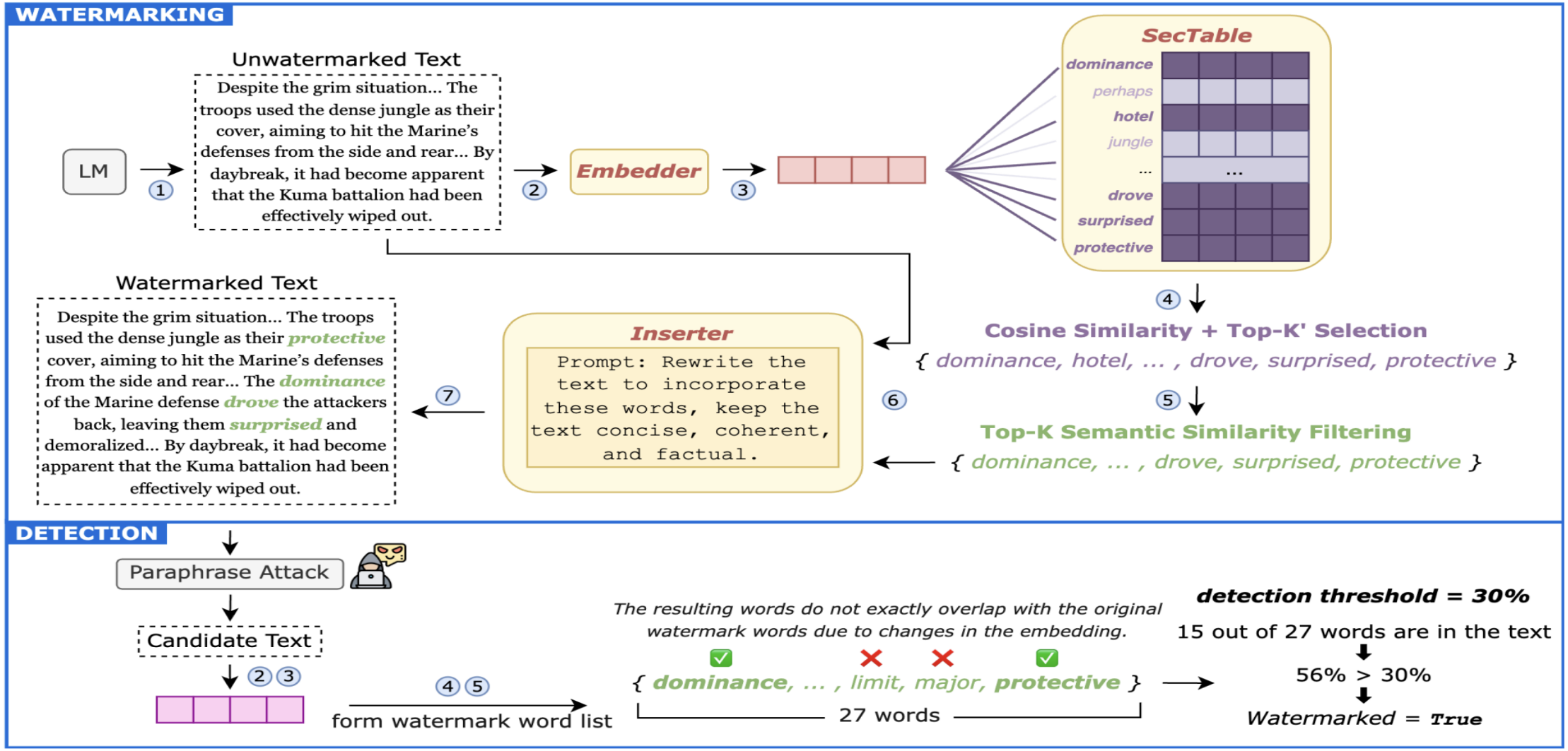
Post-Processing Watermarking

- Signature applied after text is generated
- Critical for black-box models (no access to logits)
- Achieved by rewriting or modifying the text

- **Leading Example: PostMark Algorithm**

Post-Hoc Watermarking for Robust Detection in Text Generated by Large Language Models https://aclanthology.org/2025.coling-main.364.pdf

Post-Processing Watermarking
PostMark Algorithm



POSTMARK: A Robust Blackbox Watermark for Large Language Models https://aclanthology.org/2024.emnlp-main.506.pdf

Post-Processing Watermarking
PostMark Algorithm

1. Watermark Embedding Process

1. Text Embedding: $e_t = EMBEDDER(t)$
2. .Word Selection: Pick k closest words from $SECTABLE \rightarrow Lw$ is the watermark word list

3. Use an $INSERTER$ LLM to rewrite the text to naturally include the words from L_w .

POSTMARK: A Robust Blackbox Watermark for Large Language Models
https://aclanthology.org/2024.emnlp-main.506.pdf

Post-Processing Watermarking
Watermark Detection Process

2. Watermark Detection Process

- Independently generate the *expected* word list L_w for the candidate text using the same procedure.

- Presence Score p: fraction of L_w found in text
  - Word present if $sim(w, w') \geq \theta \ (e.g., 0.7)$
  - If p exceeds threshold → text is watermarked

- Semantic design ensures robustness against paraphrasing

POSTMARK: A Robust Blackbox Watermark for Large Language Models https://aclanthology.org/2024.emnlp-main.506.pdf

Watermarking-based Approach

Pros

- **High Accuracy:** When the watermark is present, it can provide a definitive and provably accurate way to identify AI-generated content.
- **Robustness:** Well-designed watermarks, especially those based on semantics, can be difficult for an adversary to remove without significantly degrading the text's quality.
-
- **Attribution:** Watermarking can carry a payload, allowing it to embed specific information like a user ID or the name of the generating model, which is useful for tracing content origins.

Watermarking-based Approach

Cons

- **Accessibility:** Most effective watermarking methods are model-driven and need white-box access, unavailable for commercial API-based LLMs.
- **Text Quality:** Embedding watermarks can reduce fluency, quality, or factual accuracy.
- **Paraphrasing Vulnerability:** Token-based watermarks are easily broken by paraphrasing that preserves meaning.
- **Computational Cost:** Post-processing methods for black-box models are expensive, often requiring another LLM to rewrite the text.

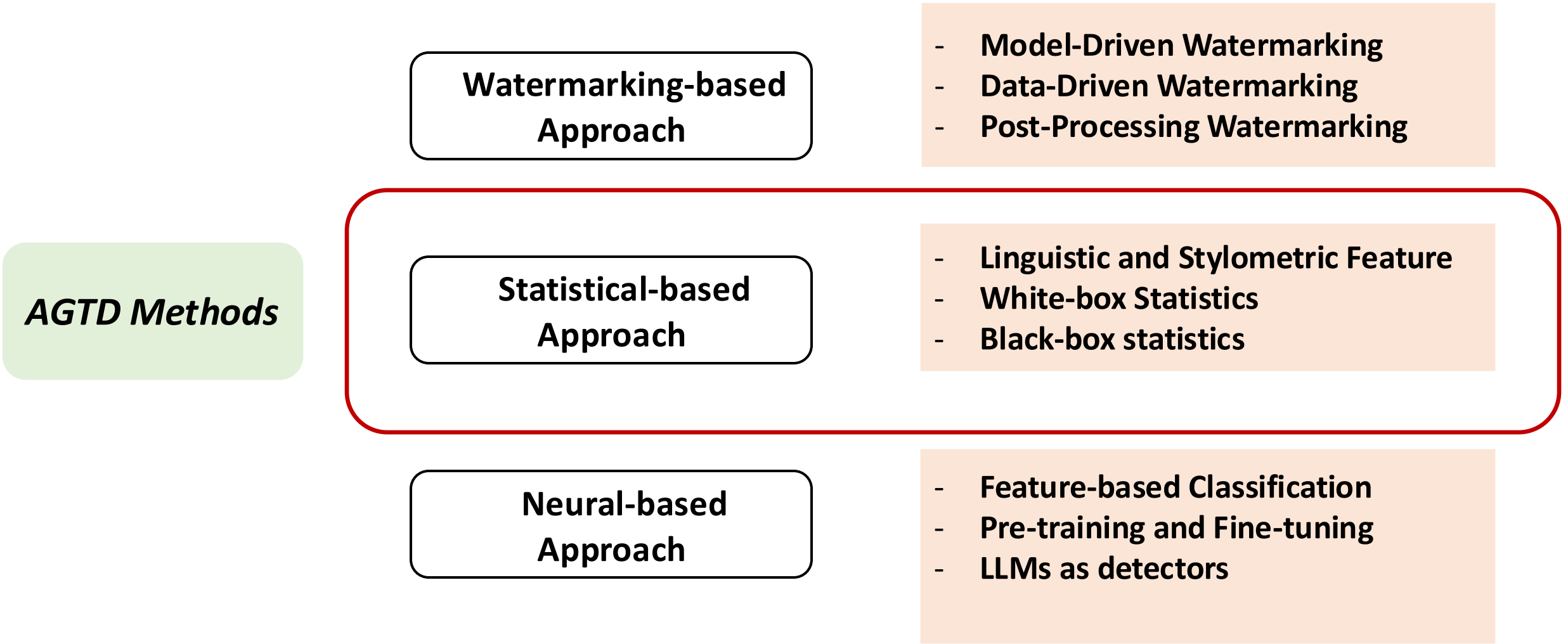
On the Reliability of Watermarks for Large Language Models

On the Reliability of Watermarks for Large Language Models
https://arxiv.org/pdf/2306.04634

https://levelup.gitconnected.com/can-we-detect-ai-generated-text-91293463dc52

https://www.youtube.com/watch?v=pJxgKdAUuXA&ab_channel=Cohere

AI-Generated Text Detection Methods taxonomy



Statistical-based Approach

Concept Overview

- Statistical detection relies on subtle artifacts in AI text
- Passive and reactive: analyze text post-generation
- No need for access to generation process (black-box)
- Early LLMs: predictable → detected via perplexity/burstiness
- RLHF improved diversity, weakening simple metrics
- Advanced approaches: curvature (DetectGPT), resampling, proxy models
- Detection arms race: as LLMs mimic humans, features become abstract and costly

Statistical-based Approach

1. **Linguistic and Stylometric Feature Analysis**
2. **White-box Statistics**
3. **Black-box Statistics**

Statistical-based Approach
Linguistic and Stylometric Feature Analysis

1. Perplexity

- Measures predictability of text under a model
- AI text often has lower perplexity (more predictable)

- Mathematical Definition:

$$PP(W) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(w_i|w_{<i})\right)$$

- Lower PP → model is confident → AI-generated
- Human text: higher PP, more varied word choice

https://research.aimultiple.com/ai-generated-text-detector/

Statistical-based Approach
Linguistic and Stylometric Feature Analysis

2. Burstiness

- Captures variation in sentence length/structure
- Human writing: bursty (short + long sentences)
- AI text: more uniform and monotonous

- Formula:

$$B = \frac{\sigma - \mu}{\sigma + \mu}$$

- $B \approx 0$ → regular pattern → AI-generated
- $B \approx 1$ → high variability → human-like

https://research.aimultiple.com/ai-generated-text-detector/

Statistical-based Approach
Linguistic and Stylometric Feature Analysis

3. N-gram Frequencies & G-test

- LLMs may bias usage of certain word sequences
- Example: overuse of phrases like 'in conclusion'
- Compare AI vs human corpora using G-test

- Formula:

$$G = 2 \sum_i O_i \cdot \ln\left(\frac{O_i}{E_i}\right)$$

- High G-score with low p-value → strong tell of AI text

https://medium.com/google-cloud/detecting-ai-generated-text-by-uncovering-its-statistical-tells-042c8d0e3a24

Statistical-based Approach

1. Linguistic and Stylometric Feature Analysis
2. **White-box Statistics**
3. Black-box Statistics

Statistical-based Approach
White-Box Detection

- Requires access to model internals (log-probabilities)
- DetectGPT is main example
- Hypothesis: AI text lies in negative curvature regions
- Perturb human vs AI text differently:
  - AI: perturbations → lower log-probability
  - Human: perturbations → mixed outcomes

DetectGPT – Perturbation Discrepancy

- Perturbation discrepancy:

$$d(x, p_\theta, q) \triangleq \log p_\theta(x) - \mathbb{E}_{\tilde{x}\sim q(\cdot|x)}[\log p_\theta(\tilde{x})]$$

- q is perturbation model (e.g., T5 for paraphrasing)
- Expectation approximated by sampling k perturbations
- High $d(x) \Rightarrow$ AI-generated
- Captures curvature of log-probability landscape

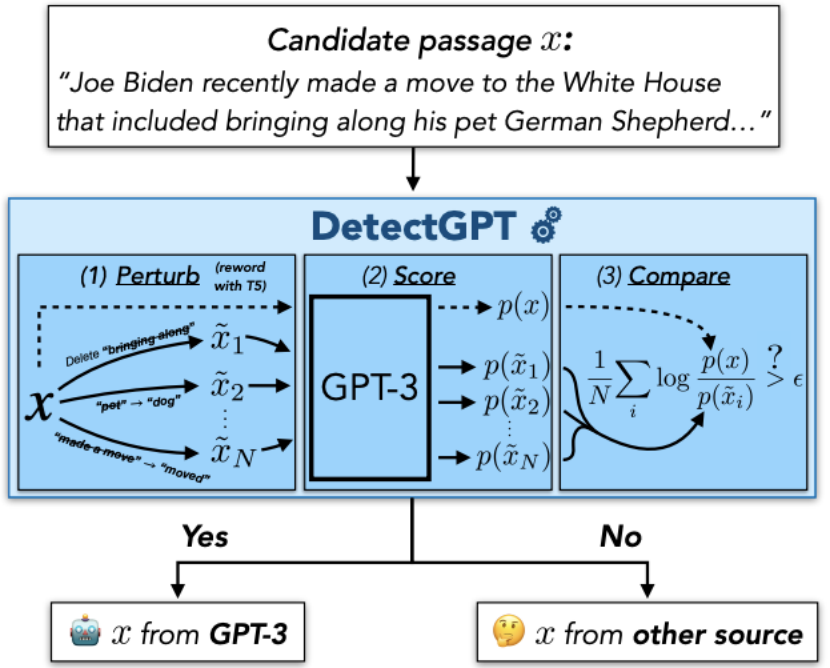


Figure 1. We aim to determine whether a piece of text was generated by a particular LLM p, such as GPT-3. To classify a candidate passage x, DetectGPT first generates minor **perturbations** of the passage $\tilde{x}_i$ using a generic pre-trained model such as T5. Then DetectGPT **compares** the log probability under p of the original sample x with each perturbed sample $\tilde{x}_i$. If the average log ratio is high, the sample is likely from the source model.

DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature https://openreview.net/pdf?id=UiAyIILXRd

Statistical-based Approach

1. Linguistic and Stylometric Feature Analysis
2. White-box Statistics
3. **Black-box Statistics**

Statistical-based Approach Resources

Black-Box Detection

- Most practical scenario: only text I/O available
- Cannot access logits
- Methods use proxy models, resampling, or statistical tests

- Examples:
  - Log-Likelihood Log-Rank Ratio (LRR)
  - Proxy-Guided Efficient Re-Sampling (POGER)
  - Zero-Shot Statistical Tests

Black-Box Detection
Log-Likelihood Log-Rank Ratio (LRR)

- Quantifies sensitivity to perturbations using proxies

- Formula

$$LRR = -\Sigma \log r_\theta(x_i|x < i) \,/\, \Sigma \log p_\theta(x_i|x < i)$$

- r_θ: rank of true token among next-token candidates
- AI text: higher LRR (chooses high-probability, high-rank tokens)
- Distinct ratio pattern vs human text

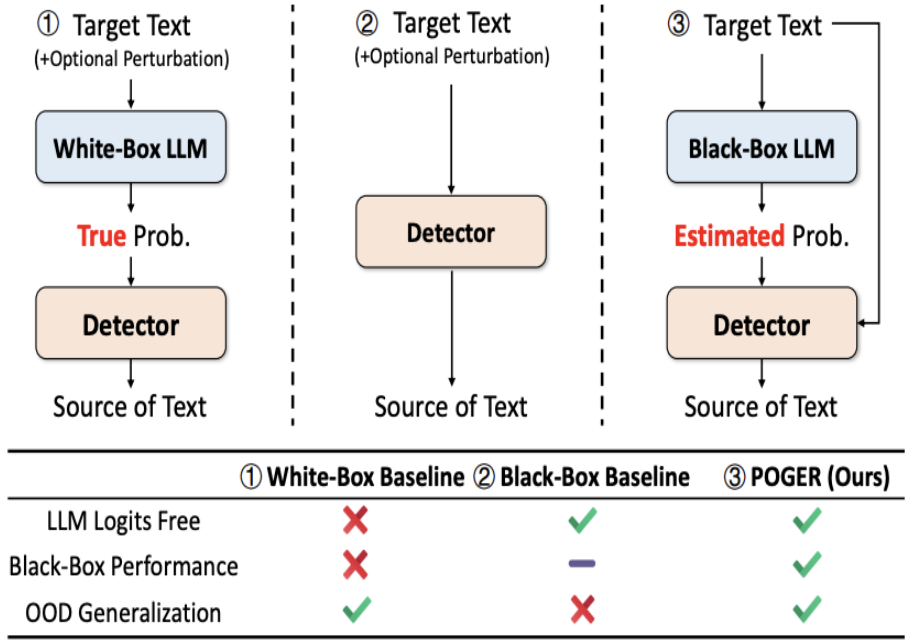
Decoding the AI Pen: Techniques and Challenges in Detecting AI-Generated Text
https://arxiv.org/abs/2403.05750

Black-Box Detection
POGER (Proxy-Guided Resampling)

- Challenge: costly to estimate probabilities via resampling

- POGER solution:
  - Use proxy model to find representative words
  - Representative = high uncertainty or distinctive usage
  - Perform resampling only at these positions

- Efficiently approximates white-box features with fewer queries



	① White-Box Baseline	② Black-Box Baseline	③ POGER (Ours)
LLM Logits Free	✗	✓	✓
Black-Box Performance	✗	–	✓
OOD Generalization	✓	✗	✓

Figure 1: Paradigm comparison between our proposed POGER and existing white-box/black-box methods. POGER does not require LLMs' internal states like output logits and performs better than the other types of baselines under black-box and out-of-distribution (OOD) settings.

Ten Words Only Still Help: Improving Black-Box AI-Generated Text Detection via Proxy-Guided Efficient Re-Sampling https://arxiv.org/pdf/2402.09199

Black-Box Detection
Zero-Shot Statistical Tests

- Theoretical guarantees in black-box settings

- Core result:

$$\log - perplexity\ \ell_A(Y_N) \approx\ cross - entropy\ \hat{\mathrm{h}}_N(B, A)$$

$$as\ N\ \rightarrow\ large$$

- If ℓ_A deviates from $\hat{\mathrm{h}}_N(A, A)$ significantly → text not from A
- Errors (Type I & II) decrease exponentially with text length
- Enables robust detection even without source model access

Zero-Shot Statistical Tests for LLM-Generated Text Detection using Finite Sample Concentration Inequalities
https://arxiv.org/pdf/2501.02406

Statistical-based Approach

Pros

- **Black-box applicability:** Works without access to model internals → usable on any text.
- **Interpretability:** Simple metrics (perplexity, burstiness) are easy to explain.
- **Zero-shot capability:** Advanced methods (e.g., DetectGPT) work without task-specific training.
- **Theoretical grounding:** Error rates decrease exponentially with longer text

Statistical-based Approach

Cons

- **Brittleness & evasion:** Fingerprints are superficial and can be erased by paraphrasing or advanced LLMs.
- **Performance vs. access trade-off:** White-box = accurate but rare; black-box = accessible but less accurate/expensive.
- **High computational cost:** Advanced methods often require heavy resampling or many API calls.
- **Bias risk:** Some detectors unfairly flag text from non-native English speakers.

AI-Generated Text Detection Methods taxonomy

AGTD Methods

Watermarking-based Approach

- **Model-Driven Watermarking**
- **Data-Driven Watermarking**
- **Post-Processing Watermarking**

Statistical-based Approach

- **Linguistic and Stylometric Feature**
- **White-box Statistics**
- **Black-box statistics**

Neural-based Approach

- **Feature-based Classification**
- **Pre-training and Fine-tuning**
- **LLMs as detectors**

Neural-Based Methods

- Neural networks (esp. transformers) dominate AIGT detection
- Detection framed as binary classification
- Fine-tuning encoders (RoBERTa, BERT) → >99% accuracy in-domain
- Weakness: poor generalization to unseen LLMs, domains, paraphrasing
- New research: interpretable and robust models (e.g., Sparse Autoencoders)
- Dilemma: SOTA accuracy vs. generalization and trustworthiness

Neural-Based Methods

1. **Feature-Based Classifiers**
2. **Pre-training Classifiers**
3. **LLMs as Detectors**

Neural-Based Methods

Feature-Based Classification Models

- Pipeline: Text -> Feature Extractor -> Classifier -> {Human, AI}

- Types of Features:
  - Linguistic & stylometric: readability, type-token ratio, POS distribution
  - Embedding-based: extract embeddings (e.g., RoBERTa [CLS]) → MLP/XGBoost
  - Interpretable neural features: Sparse Autoencoders (SAEs)

- Decompose activations into semantic/stylistic features
- Enable explainable predictions

- Classifiers: SVM, XGBoost, or neural networks
- Rich feature sets + XGBoost = competitive SOTA

Neural-Based Methods
Pre-training & Fine-tuning Paradigm

- Transfer learning pipeline:
    1. Select pre-trained backbone (BERT, RoBERTa, DeBERTa)
    2. Add classification head (linear layer + softmax)
    3. Fine-tune on labeled dataset (human vs AI text)

- Loss function (Binary Cross-Entropy):

- Pros: very high accuracy on benchmarks
- Cons: overfits to artifacts, poor OOD robustness

Neural-Based Methods
LLMs as Introspective Detectors

- Use LLMs' own behavior for detection

- Rewriting-based detection:
  - LLM rewrites candidate text
  - AI text: minimal changes (already aligned)
  - Human text: larger changes
  - Compare original vs rewritten (semantic similarity, edit distance)

- Advantage: generalizable, no dedicated classifier needed

Detecting AI-generated text via machine-based rewriting
https://inventions.techventures.columbia.edu/technologies/detecting-ai--CU24139

Neural-Based Methods
Theoretical Limits of Detection

- Detection bounded by distribution distance between human & AI text
- Total Variation (TV) Distance:

$$TV(M,H) = 1/2 \, \Sigma \, |M(x) - H(x)|$$

0 → identical distributions
1 → disjoint distributions

- AUROC bound:

$$AUROC(D) \leq 1/2 + TV(M,H)$$

- As AI text ≈ human text:
  - TV → 0
  - Max AUROC → 0.5 (random guessing)

- Implication: Perfect undetectability may be possible

Can AI-Generated Text be Reliably Detected?
https://arxiv.org/abs/2303.11156

Neural-Based Methods

Pros

- **High accuracy:** Fine-tuned neural classifiers can exceed 99% on in-domain benchmarks.
- **Feature learning:** They automatically capture subtle patterns beyond predefined statistical features.
- **Fast inference:** After training, predictions are quick, enabling real-time use.
- **Versatile:** Architectures adapt to tasks from binary classification to stylistic analysis.

Neural-Based Methods

Cons

- **Poor generalization:** High accuracy often collapses on out-of-distribution data, as models overfit training quirks.
- **Adversarial vulnerability:** Paraphrasing or “AI humanizer” tools can bypass detection easily.
- **High training cost:** Requires large labeled datasets and heavy compute.
- **Black-box nature:** Lack of interpretability limits trust in high-stakes use.
- **Bias risk:** Can unfairly flag non-native English text as AI-generated.
- **Maintenance needs:** Frequent retraining is required to keep up with new LLMs.

Part 3: Benchmarks and Open Challenges

Key Datasets for AIGT Detection

- GPT-2 Output Dataset – 500k+ samples, first large-scale benchmark
- GROVER – News domain, Grover-Mega generator
- HC3 / HC3-zh / HC3 Plus – Human vs ChatGPT Q&A (English & Chinese)
- CHEAT – Academic abstracts, refined & blended to simulate misuse
- ArguGPT – Academic/argumentative essays across GPT models
- DeepfakeTextDetect – 27 LLMs, multiple genres including GPT-4 & paraphrases
- weepFake – Twitter-based human vs GPT/RNN text

https://aclanthology.org/2025.cl-1.8.pdf

Evaluation Benchmarks

- TuringBench – Early benchmark (GPT-1 to GPT-3, Grover, CTRL, etc.)
- MGTBench – Modular, includes multiple LLMs (ChatGPT, GPT4All, StableLM)
- GPA Benchmark – 600k academic texts: human, GPT-written, completed, polished
- Scientific Articles Benchmark – ArXiv human vs GPT/ChatGPT/Galactica
- MULTITuDE – 11 languages, 8 LLMs, multilingual focus
- HANSEN – Spoken text benchmark (ChatGPT, PaLM2, Vicuna)
- M4 – Large multilingual, multi-domain benchmark (ChatGPT, BLOOMz, LLaMA)
- DetectRL – Real-world scenarios, adversarial robustness, 230k+ samples

https://aclanthology.org/2025.cl-1.8.pdf

Shared Tasks and Competitions

- ARATECT @ArabicNLP’2025
- M-DAIGT @ RANLP (2025)
- PAN @ CLEF (2023–2025): Multilingual AI-generated text detection
- SemEval 2024: Multilingual news & opinion detection
- GenAI @ BEA 2024–25: Educational domain (student essays)
- DAIGT Competitions (Kaggle): Diverse, real-world web/social data
- WMT Tasks: Human vs machine translation overlaps

- **Trend**: Expansion into multilingual, cross-domain, academic, and social media

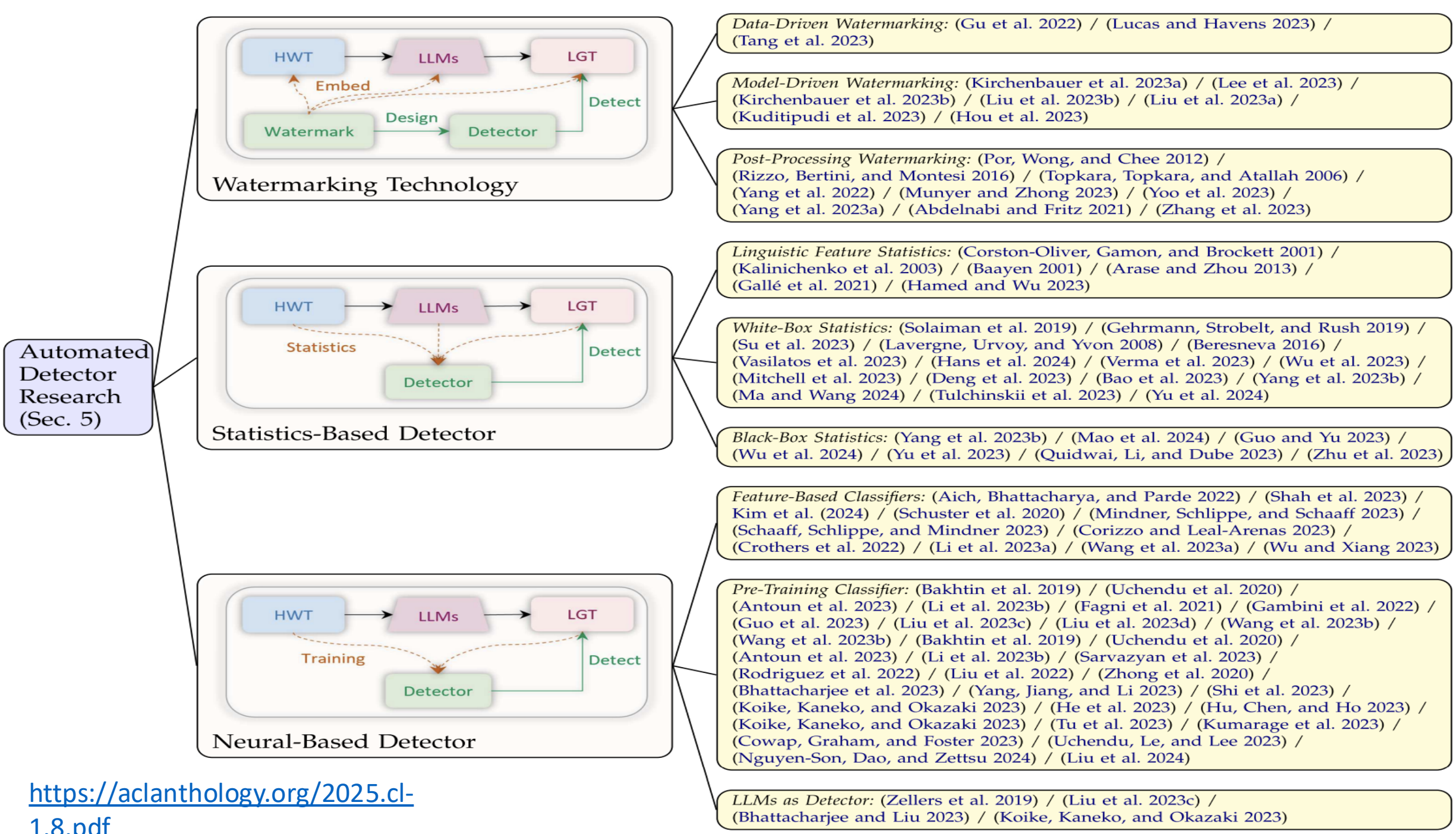
Open Challenges (1/2)

- Out-of-Distribution Generalization – detectors overfit to training domains
- Adversarial Robustness – paraphrasing, prompt attacks, AI humanizers
- Multilingual Gaps – English dominance, few low-resource corpora
- Evaluation Metrics – beyond accuracy/F1: need calibration, AUROC, uncertainty

Open Challenges (2/2)

- Bias & Fairness – detectors flagging non-native English as AI text
- Computational Cost – white-box and resampling methods are heavy
- Lack of Interpretability – neural methods often black-box
- Theoretical Limits – TV distance shows AUROC upper bound → 0.5 as LLMs improve
- **Future Research:** hybrid watermark + neural, multilingual corpora, explainable models

Wrap-Up & Discussion



https://aclanthology.org/2025.cl-1.8.pdf

Resources

- The Imitation Game revisited: A comprehensive survey on recent advances in AI-generated text detection https://www.sciencedirect.com/science/article/pii/S0957417425003161
- A Survey of AI-generated Text Forensic Systems: Detection, Attribution, and Characterization https://arxiv.org/pdf/2403.01152
- A Survey on Detection of LLMs-Generated Content https://aclanthology.org/2024.findings-emnlp.572.pdf
- A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions https://aclanthology.org/2025.cl-1.8.pdf
- A Survey on the Possibilities & Impossibilities of AI-generated Text Detection https://openreview.net/pdf?id=AXtFeYjboj

- https://github.com/Nicozwy/AIGTD-Survey

Resources

- Does DETECTGPT Fully Utilize Perturbation? Bridging Selective Perturbation to Fine-tuned Contrastive Learning Detector would be Better https://aclanthology.org/2024.acl-long.103.pdf

- A Survey of Text Watermarking in the Era of Large Language Models https://arxiv.org/pdf/2312.07913

- Watermarking for AI Content Detection: A Review on Text, Visual, and Audio Modalities https://arxiv.org/pdf/2504.03765v1

- Watermarking Language Models through Language Models https://arxiv.org/pdf/2411.05091v1

Thank You!