Fast-DetectGPT

Understanding sampling-based zero-shot AI-generated Text Detection

Krystof Bobek
Mara Dragomir
Mahdi Rahimi
Jose Garcia
Salvador Torpes

Table of Contents

Introduction

The Task

Classifying input text as human or AI-generated

With roots in blues rock and psychedelic rock, the bands that created heavy metal developed a thick, massive sound, characterized by highly amplified distortion, extended guitar solos, emphatic beats, and overall loudness.

Which example is human and which is generated by a LLM?

Metal music is a genre of rock music that typically features heavy, distorted guitar riffs and fast-paced tempos. It often incorporates elements of other musical genres such as punk, hardcore, and classical music, and is known for its aggressive, energetic sound.

The Task

Classifying input text as human or AI-generated

Human

With roots in blues rock and psychedelic rock, the bands that created heavy metal developed a thick, massive sound, characterized by highly amplified distortion, extended guitar solos, emphatic beats, and overall loudness.

LLM

Metal music is a genre of rock music that typically features heavy, distorted guitar riffs and fast-paced tempos. It often incorporates elements of other musical genres such as punk, hardcore, and classical music, and is known for its aggressive, energetic sound.

Difficult task for humans, but maybe not so much for AI

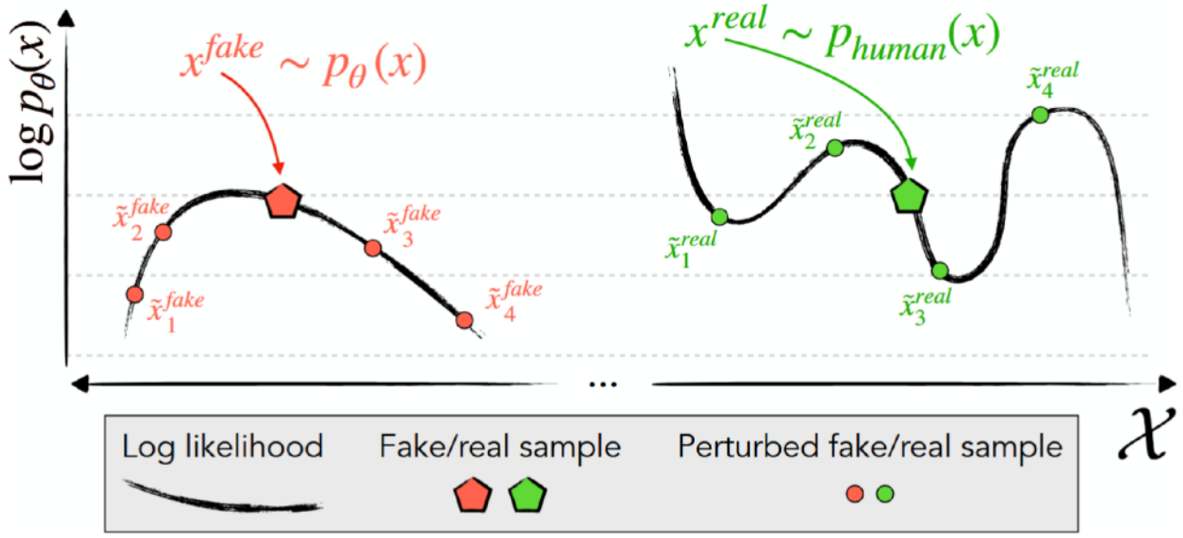
Motivation

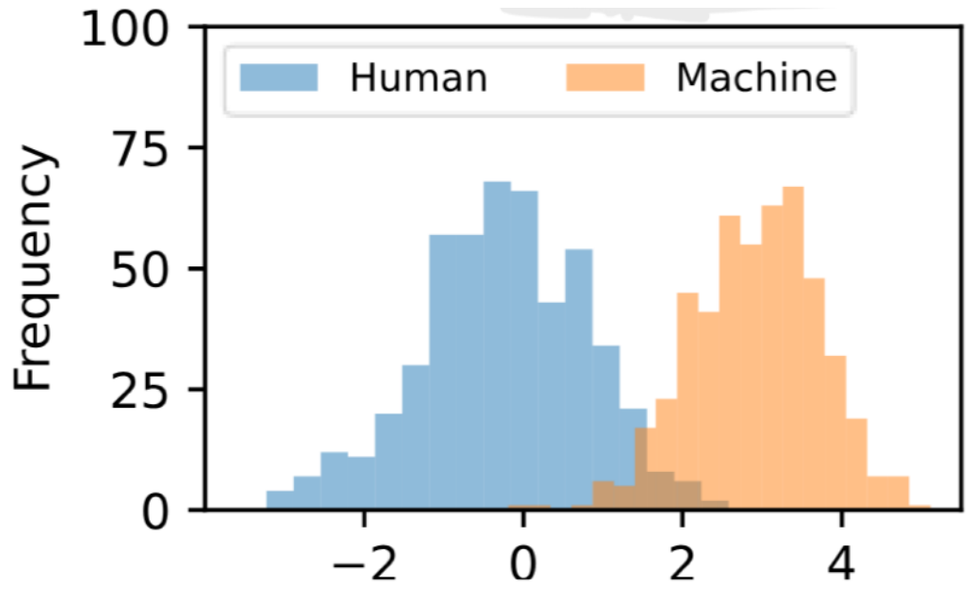
Why is it important to detect AI-generated content?

- Preventing mis- and disinformation
- Preventing plagiarism
- Transparency
- Preserving human creativity and intellect

The Idea

DetectGPT & Fast DetectGPT





The probability curvature of a text sample fed to a model can show whether the text is written by a human or by the model itself

Methodology

Models

Models

FastDetectGPT uses three different Large Language Models in its pipeline:

Source Model

- Used for dataset creation
- Takes a collection of human sentences as input, and generates a corresponding AI sentence for each using the first 30 tokens of the sentence

Sampling Model

- Used for perturbing the input sentences
- Samples from the probability distribution of tokens over each position in the sentence

Scoring Model

- Used for the probability curvature
- Uses the probability distribution of tokens over each position in the sentence to compute the loss
- Classifies the sentence via a threshold

Settings

The experiments of FastDetectGPT can be done under a black box or a white box setting.

Black Box

where the source and scoring models can be different (the model doesn't know what model generated the text)

(Our Focus)

White Box

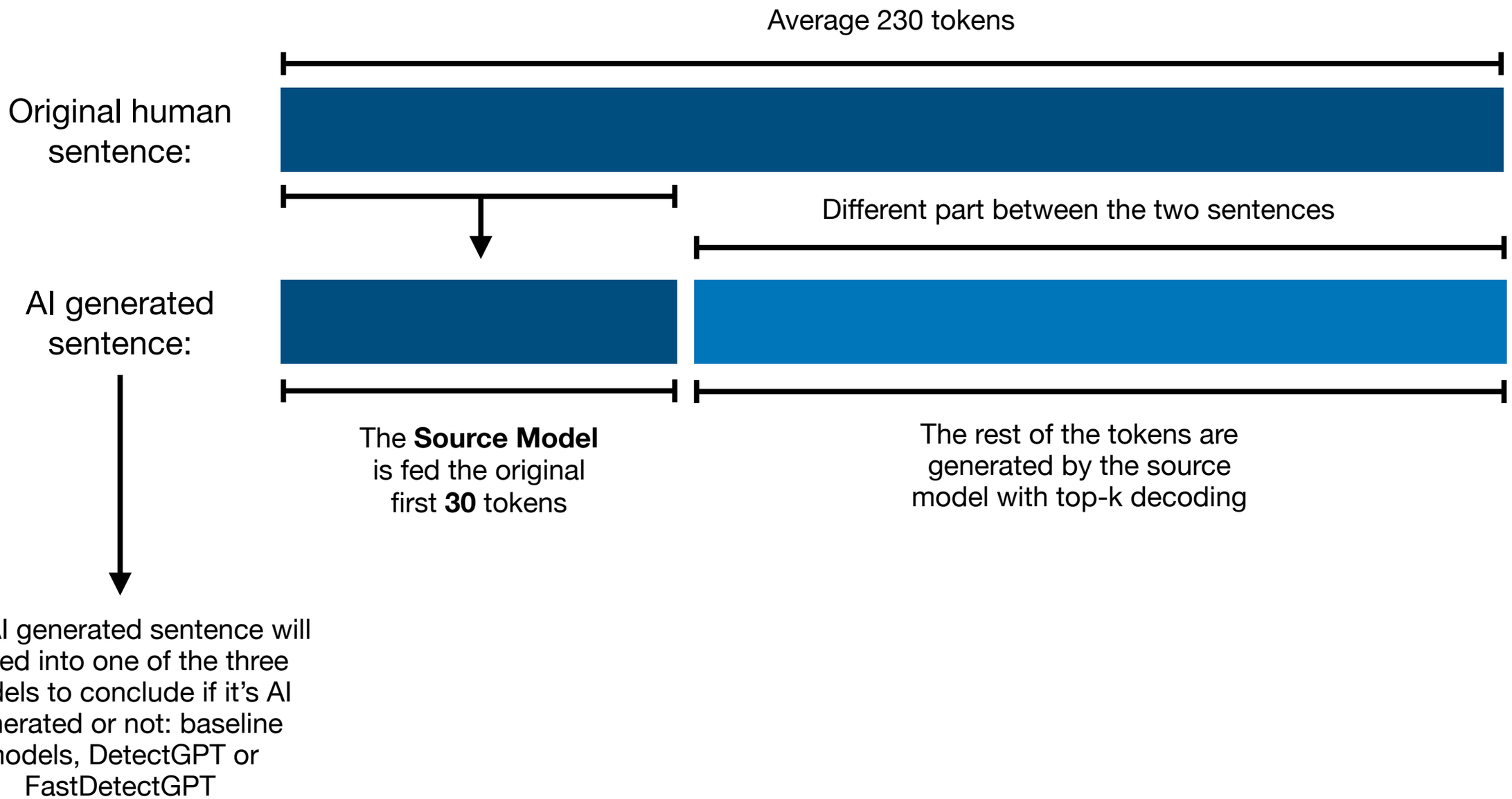
where the source and scoring models are the same (the model classifies whether a text is generated by itself)

The sampling and scoring model may differ and that might lead to performance changes

Dataset Generation

Generating AI text

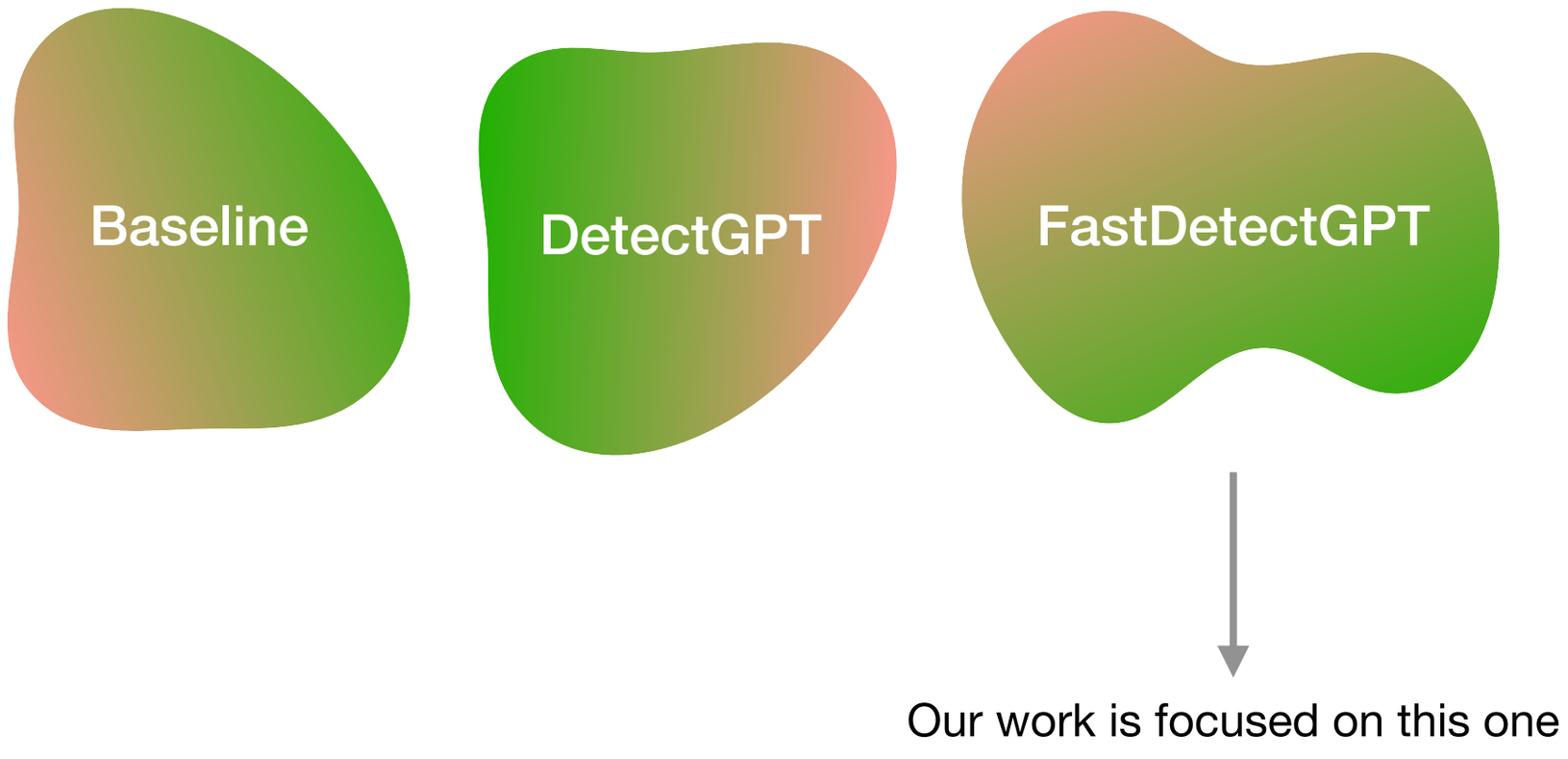
Considering we have a human text dataset with multiple entries, building the AI text will consist of building an AI equivalent sentence for each entry, as follows:



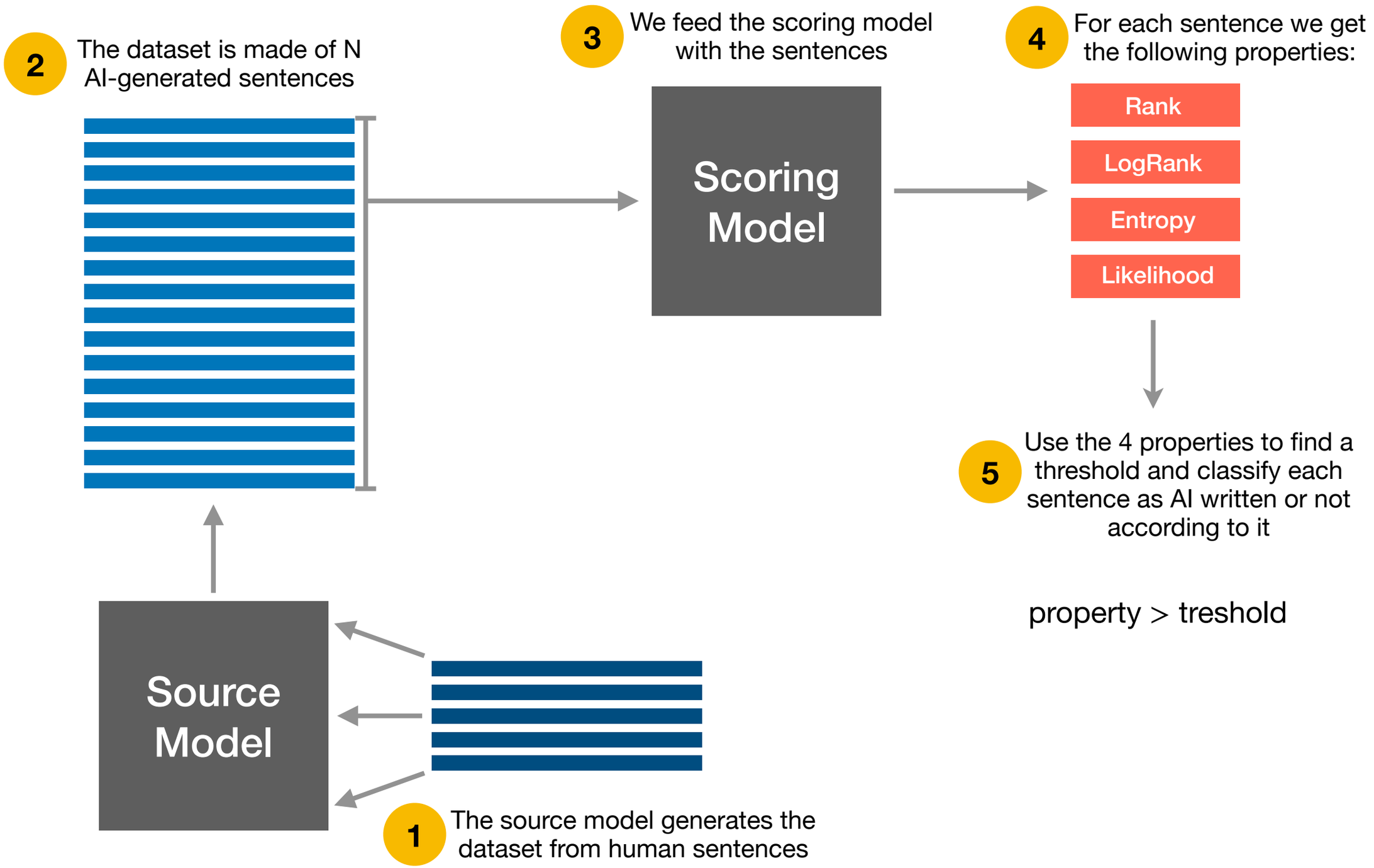
Methods

Three Different Methods for Detection

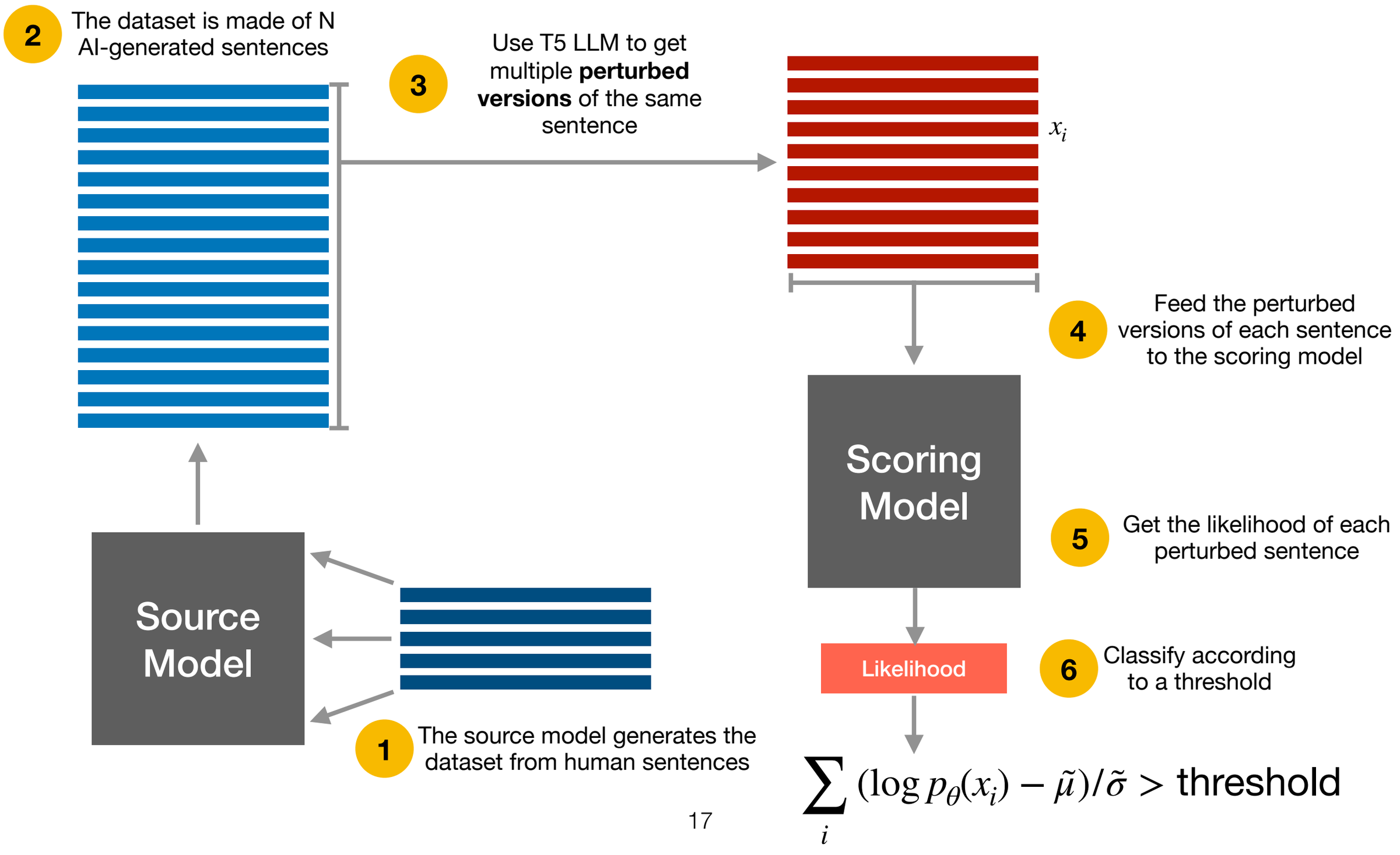
We have studied three different methods for identifying AI-written text:



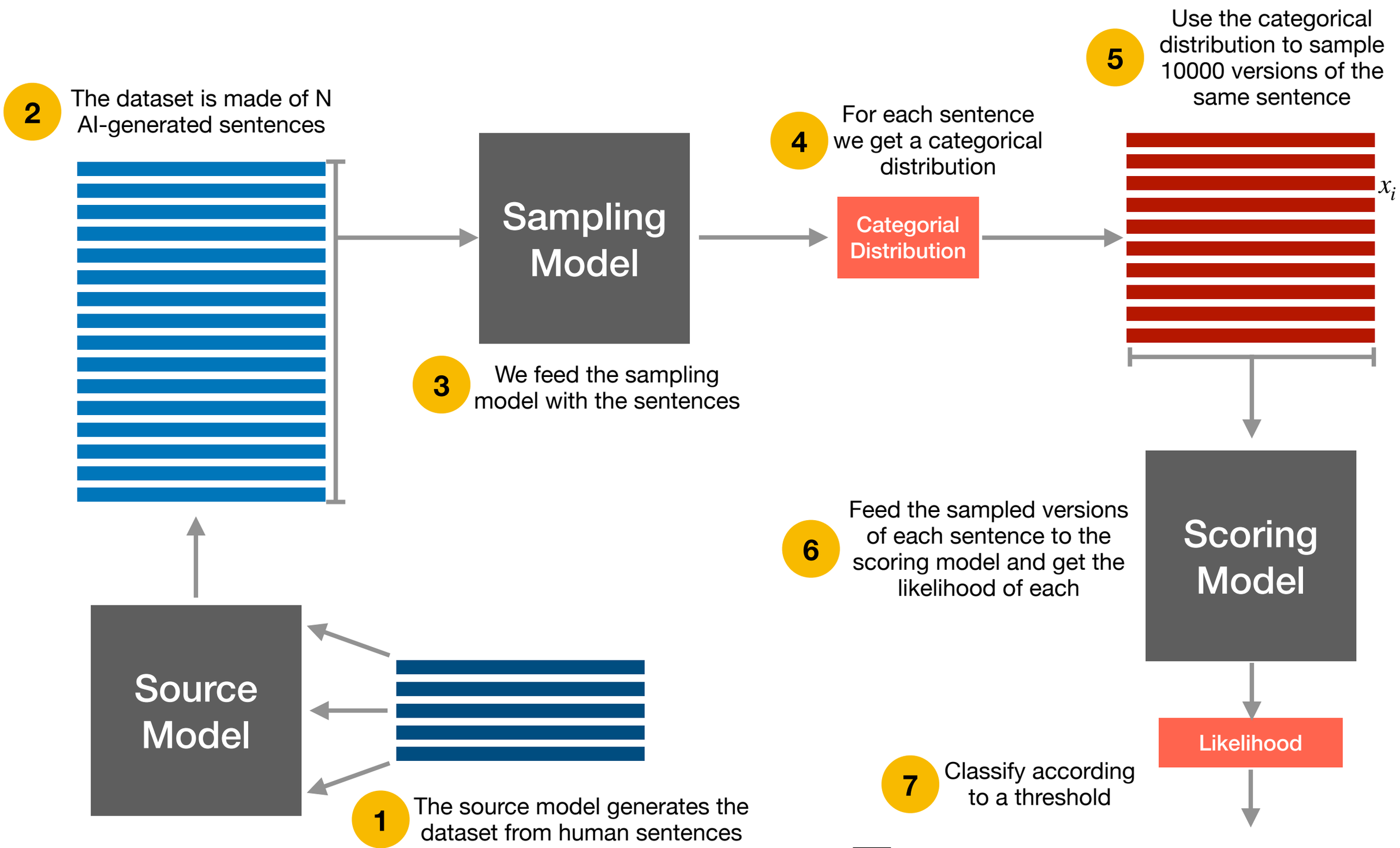
Baseline Methods



DetectGPT


2
The dataset is made of N AI-generated sentences
3
Use T5 LLM to get multiple **perturbed versions** of the same sentence
x_i
4
Feed the perturbed versions of each sentence to the scoring model
Scoring Model
5
Get the likelihood of each perturbed sentence
Likelihood
6
Classify according to a threshold
$\sum_i (\log p_\theta(x_i) - \tilde{\mu})/\tilde{\sigma} > \text{threshold}$
Source Model
1
The source model generates the dataset from human sentences

FastDetectGPT



$$\sum_i (\log p_\theta(x_i) - \tilde{\mu})/\tilde{\sigma} > \text{threshold}$$

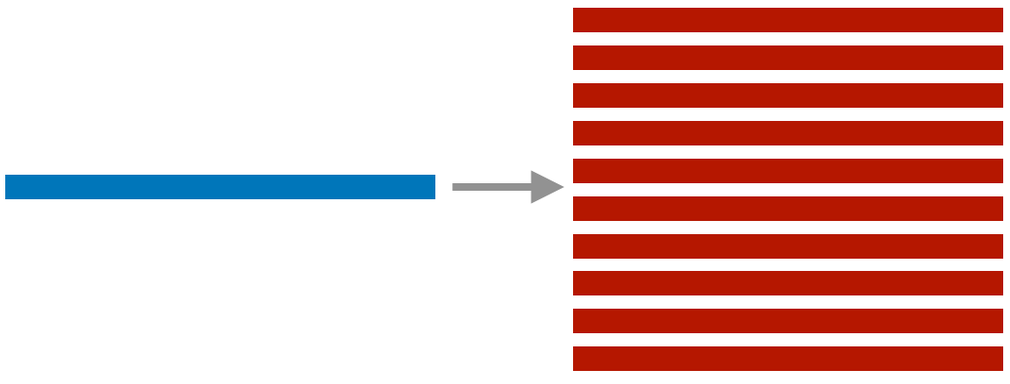
FastDetectGPT

Why are we using this log-likelihood based sum?

$\log p_\theta(x_i)$ Is the log likelihood of the sentence x_i under the scoring model (sum of each token's log probability)

$\tilde{\mu}$ Is the mean of the log likelihood of each of the 10000 sample x_i

$\tilde{\sigma}$ Is the standard deviation of the log likelihood of each of the 10000 sample x_i



1 AI generated sentence from the dataset originates **10000** samples and with those samples we compute this

$$\sum_i (\log p_\theta(x_i) - \tilde{\mu})/\tilde{\sigma} > \text{threshold}$$

The higher this sum is, the more likely it is for the sentence to be generated by an AI model since the model says the token sequence is highly likely

FastDetectGPT

Here follows the pseudo-code for FastDetectGPT:

Algorithm 1 Fast-DetectGPT machine-generated text detection.

Input: passage x, sampling model q_φ, scoring model p_θ, and decision threshold ϵ
Output: True – probably machine-generated, False – probably human-written.

function FASTDETECTGPT(x, q_φ, p_θ)
  $\tilde{x}_i \sim q_\varphi(\tilde{x}|x), i \in [1..N]$ ▷ Conditional sampling
  $\tilde{\mu} \leftarrow \frac{1}{N}\sum_i \log p_\theta(\tilde{x}_i|x)$ ▷ Estimate the mean
  $\tilde{\sigma}^2 \leftarrow \frac{1}{N-1}\sum_i (\log p_\theta(\tilde{x}_i|x) - \tilde{\mu})^2$ ▷ Estimate the variance
  $\hat{\mathrm{d}}_x \leftarrow (\log p_\theta(x) - \tilde{\mu})/\tilde{\sigma}$ ▷ Estimate conditional probability curvature
  return $\hat{\mathrm{d}}_x > \epsilon$
end function

$$\sum_i (\log p_\theta(x_i) - \tilde{\mu})/\tilde{\sigma} > \text{threshold}$$

Sampling and Scoring

Sampling and Scoring

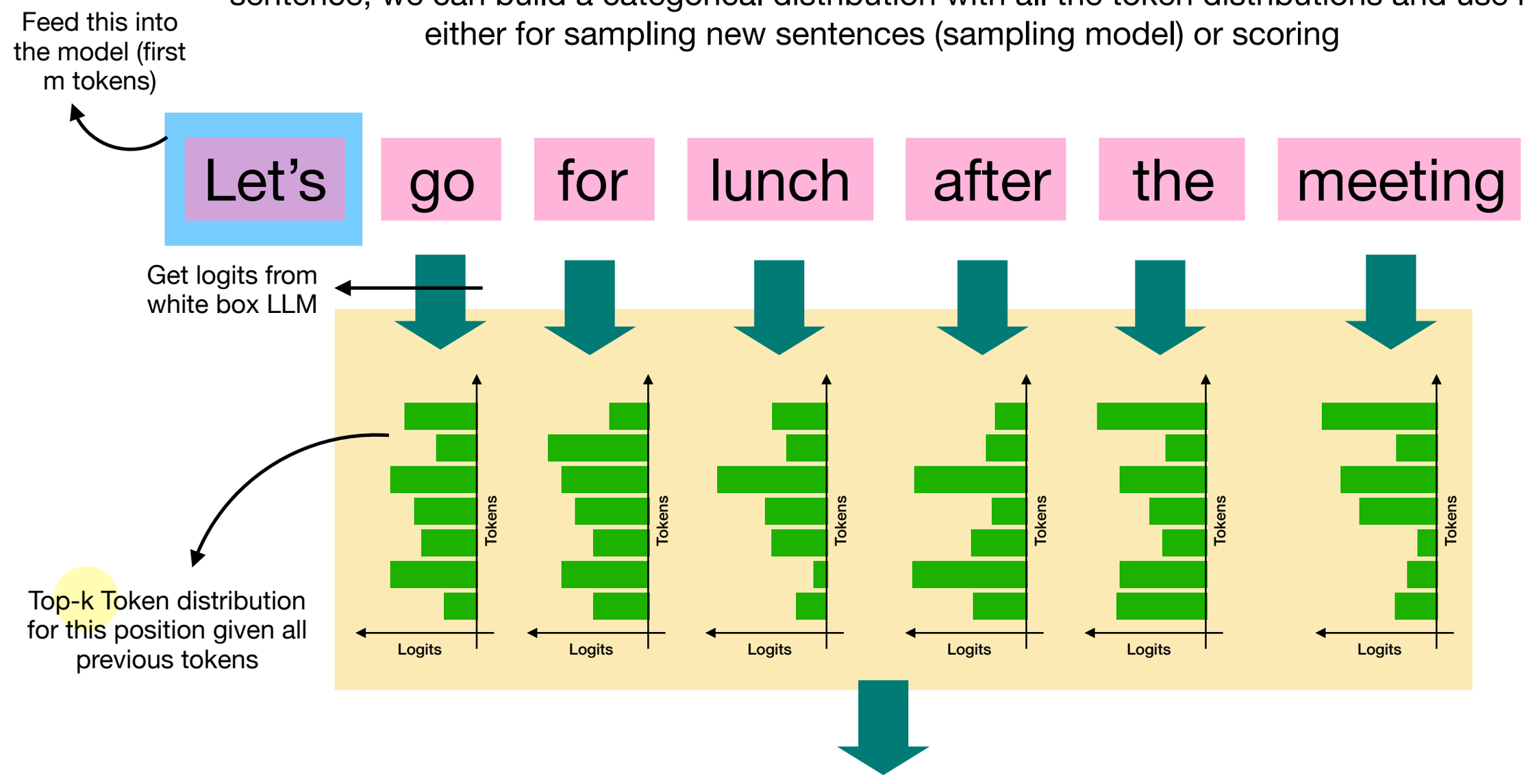
Both sampling and scoring require the token distribution at each position in the sentence - sampling for alternative token selection, scoring for calculation of conditional probability curvature at each token

Generates alternative sequences by sampling from the model’s token distribution

Scoring model used to compute conditional-probability curvature at each token; higher values mean the sequence is more likely machine-generated

Obtaining token distributions

We feed an LLM model the first m tokens of the original sentence in order to get logits for the next positions. After we have a probability distribution over each position on the sentence, we can build a categorical distribution with all the token distributions and use it either for sampling new sentences (sampling model) or scoring



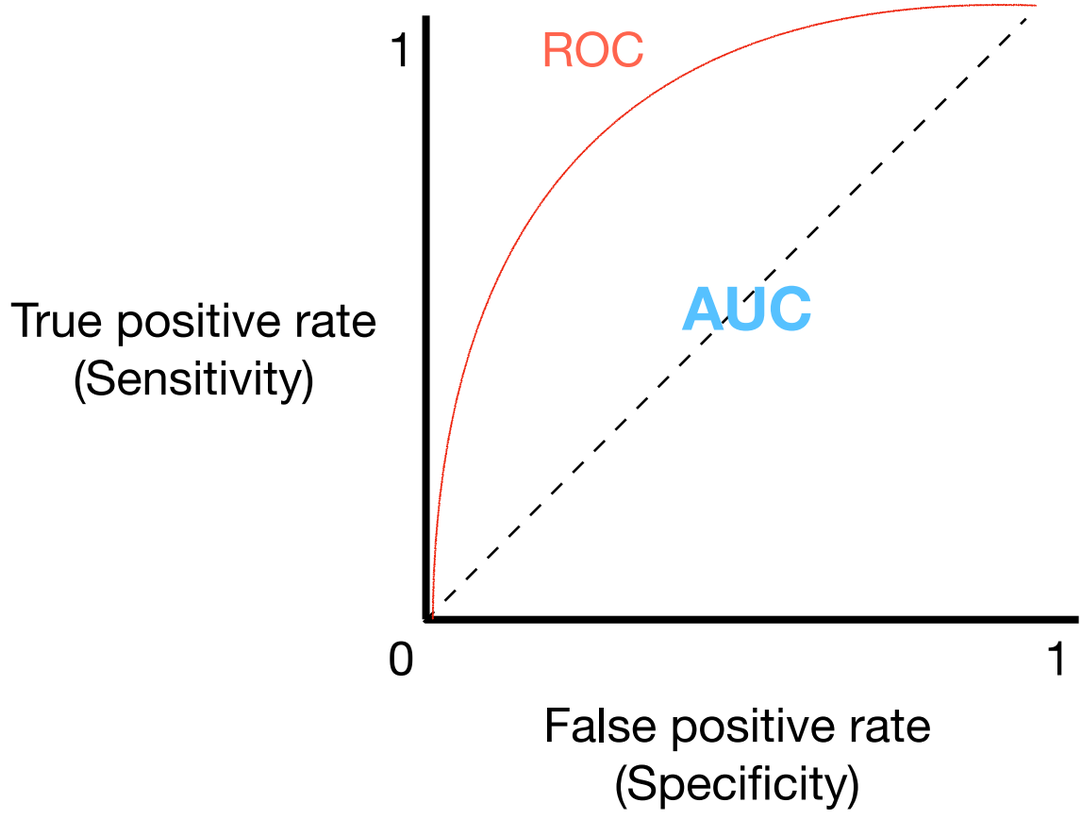
Build a categorical distribution with all token distributions for the sampling model and for the scoring model

The process of obtaining the tokens distribution for each position is done twice*: once for the Sampling Model and once for the Scoring Model

*Except for white box setting where both the models are identical.

Main Metric

Area Under Receiver Operating Curve



Experiments and Results

New Models and Datasets

Reproducibility Study

Small Open-Weight Models:

- DeepSeek R1 Distill Llama
- Microsoft Phi 2
- Mistral 7B Instruct v0.2

Metric: AUROC

Note: DetectGPT could only be used one time due to computational constraints

Method	XSum			
	R1-8B	Phi-2	Mistral	Avg.
Likelihood	0.9999	0.8782	0.9665	0.9482
Entropy	0.1645	0.5411	0.4558	0.3871
LogRank	**1.0000**	0.9051	0.9596	0.9549
DetectGPT	0.8560	-	-	-
★ Fast-DetectGPT	**1.0000**	**0.9769**	**0.9989**	**0.9919**

Method	HC3			
	R1-8B	Phi-2	Mistral	Avg.
Likelihood	0.8391	0.7139	0.7486	0.7672
Entropy	0.3521	0.4358	0.4235	0.4038
LogRank	0.8275	0.7224	0.7393	0.7631
★ Fast-DetectGPT	**0.9710**	**0.8347**	**0.8963**	**0.9007**

Speed Comparison

Fast-DetectGPT is

118.9x

faster than DetectGPT

on XSum with DeepSeek R1 Distill Llama

Close-Weight API Model

Method	GPT 4o mini		
	XSum ★	HC3	Avg.
Fast-DetectGPT	**0.8051**	0.6026	0.7039

- Lower performance due to top-20 probability restriction

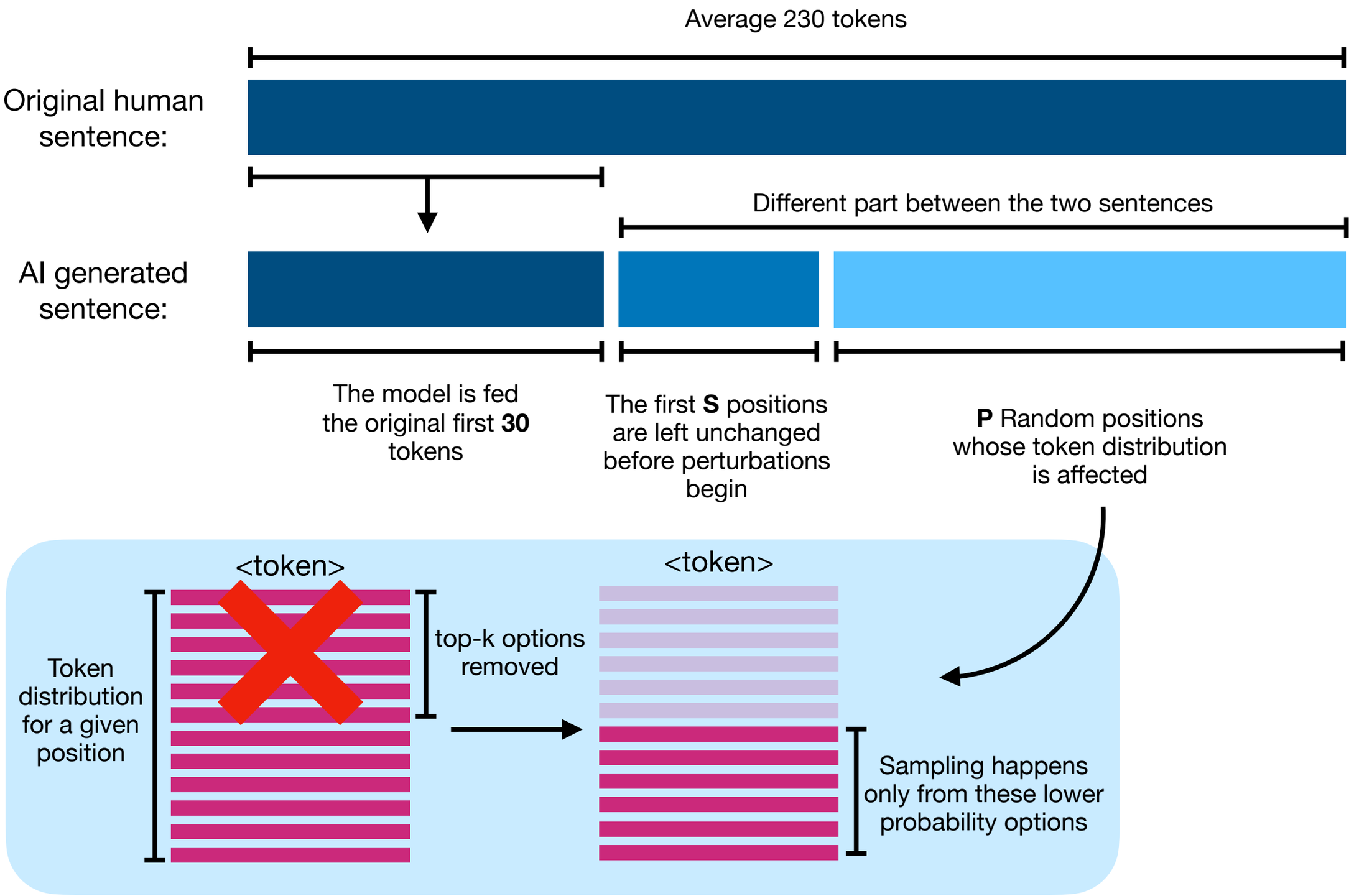
Low-Resource Language

Method	Underrepresented language			
	R1-8B	Phi-2	Mistral	Avg.
Likelihood	**1.0000**	0.9976	0.9972	0.9983
Entropy	0.0010	0.0968	0.2730	0.1236
★ LogRank	**1.0000**	**0.9984**	0.9974	**0.9986**
Fast-DetectGPT	0.9994	0.9825	**0.9995**	0.9938

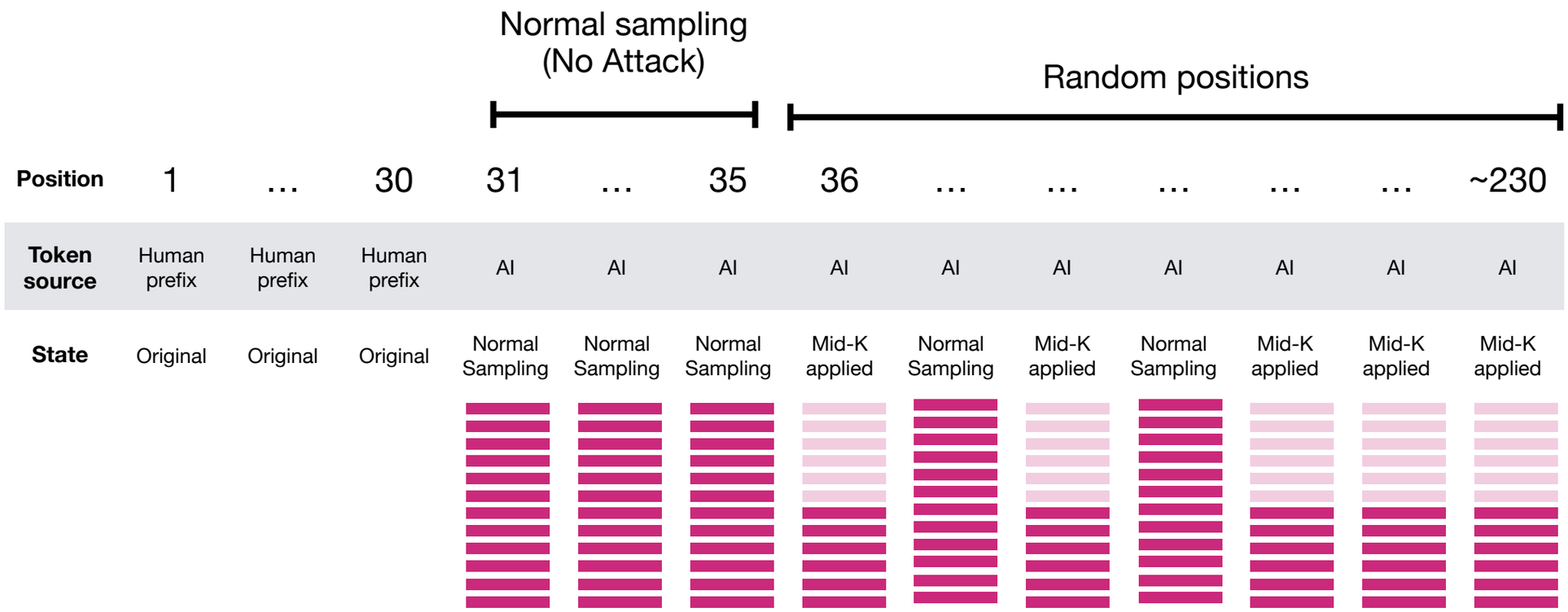
+ Competitive scores, overall robustness

Mid-K Extension

How Mid-K Works

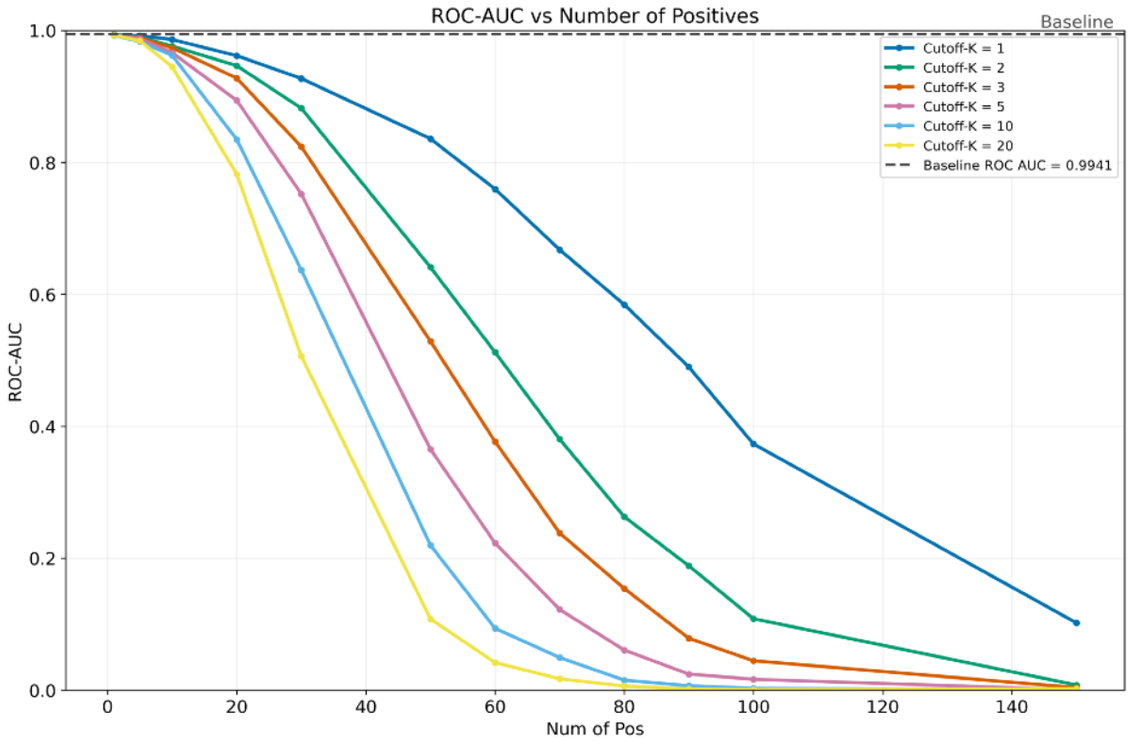


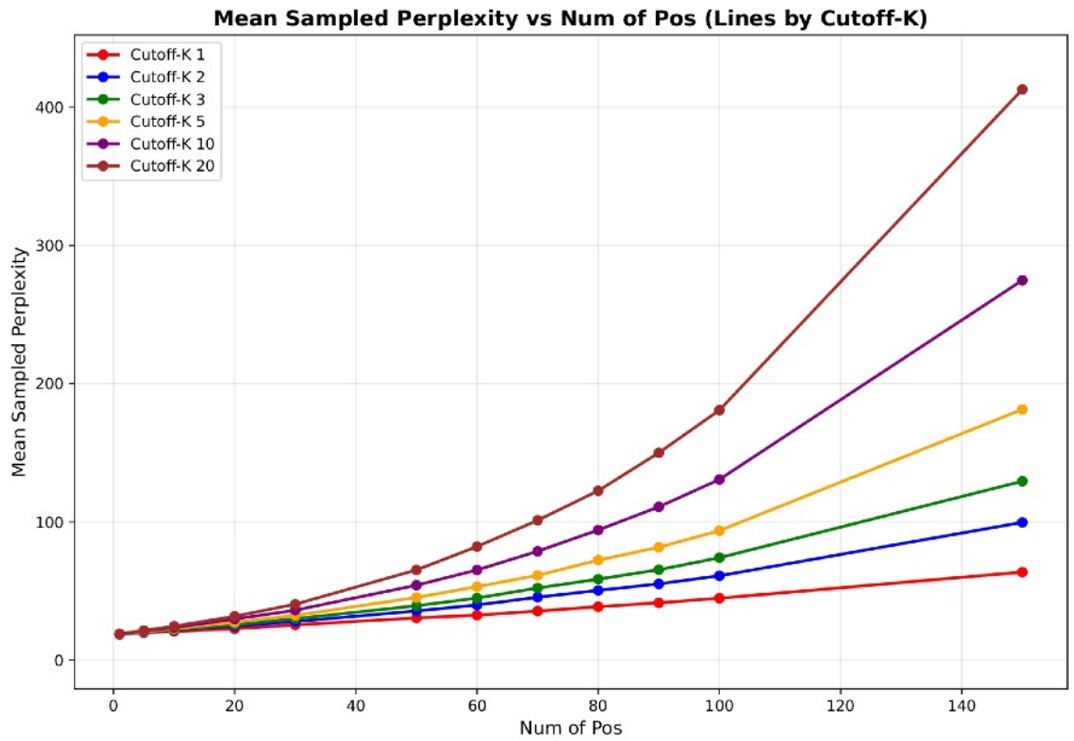
Mid-K in Action



- Key parameters:
  - **P**: Number of random positions where attack is applied
  - **K**: Number of top tokens removed at each position
  - **S**: Start position (attack begins after S tokens)

Effect of Mid-K Attack





Setup:

- Model: GPT-2
- Dataset: XSum
- Parameters: P ∈ {1, 5, 10, 20, 30, 50, 60, 70, 80, 90, 100, 150}
- K ∈ {1, 2, 5, 10, 20}, S = 5

Qualitative Examples

Fixed part: "The false message said: "Breaking: Two Explosions..."

Human	**in the White House and Barack Obama is injured." US markets were spooked by the tweet**
AI (No Attack)	**At University of Michigan Student Residences, Campus Police Service (APS) has confirmed at least one person was hurt.**
Mild (P=60, K=2)	**At University of Michigan Student Residences, Campus Police Service (APS) has confirmed at 7:18 a.b.'s report a report by Campus Police that three occupants were shot dead**
Wide (P=150, K=1)	**At University of Michigan Student Residences, Campus Fires. We apologize if we didn't find the wrong tweet. The University has since been informed.**
Strong (P=150, K=20)	**are Found at a School; Parents Support Public Alert: Close Education Week Hours Soon With Two Public Works, Infrastructure Flashed". http@worldc.en**

Conclusion

Takeaways

Fast-DetectGPT:

- Outperforms baselines on newly tested models ✓
- Two orders of magnitude faster than DetectGPT (not 340 times though) ✓
- Vulnerable towards adversarial attacks ✗
- Performance depends on full logits distribution availability ✗

Thank you!