Watermarked Sequence Length Code Generation for Large Language Model

Jongsoo Ha, Xiaohui Liang, Youxiang Zhu, Jim Schwoebel

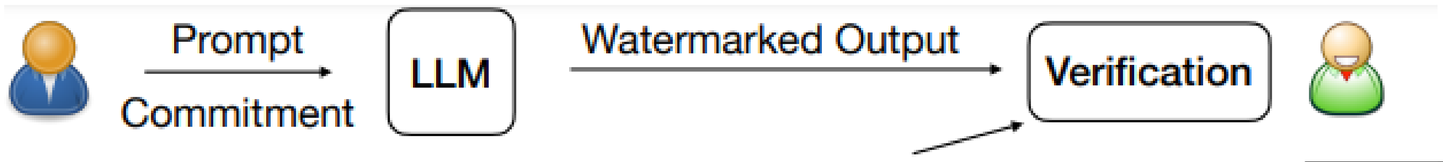
UMass Boston

ORACLE

1.Introduction

As large language models (LLMs) become widely used in code generation and content creation, **ensuring transparency and accountability is crucial**. A major challenge is distinguishing between human and AI-generated content, which raises concerns about **authenticity, ownership, and misuse**. To address this, watermarking techniques have been developed to embed hidden patterns in AI-generated text and code, allowing for traceability without compromising quality

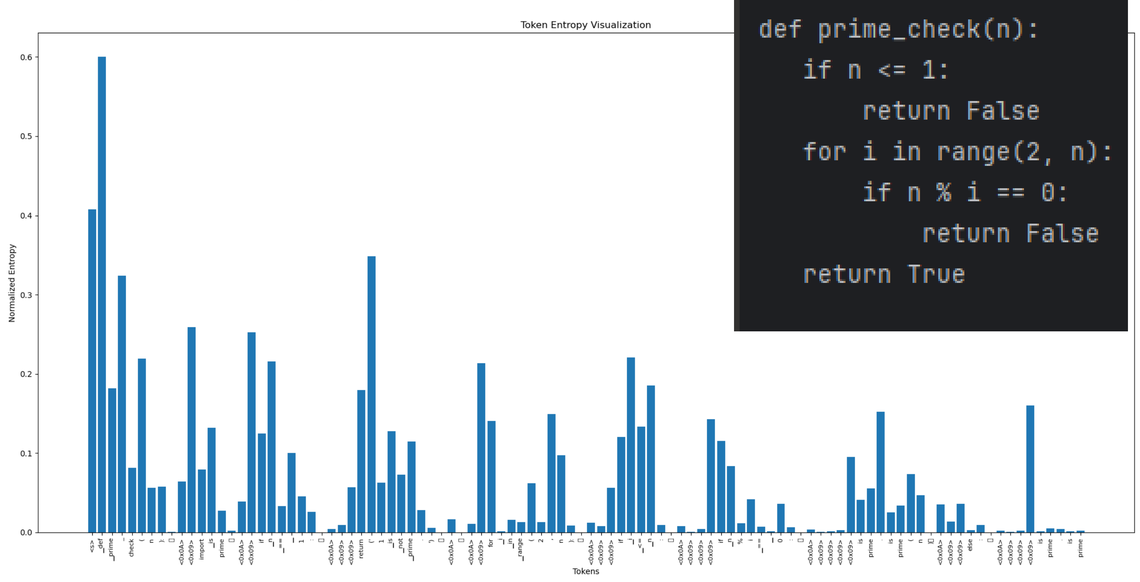
2. What is the Current limit of watermarking system?



The current watermarking system, originally **designed for essays and paragraphs**, faces **challenges when applied to code generation**. It struggles to maintain performance and exhibits a low detection rate, especially when the generated code functions correctly.

We introduce Sequence Length Watermark(**SLW**) to ensure robustness and code performance!

Entropy chart for each tokens

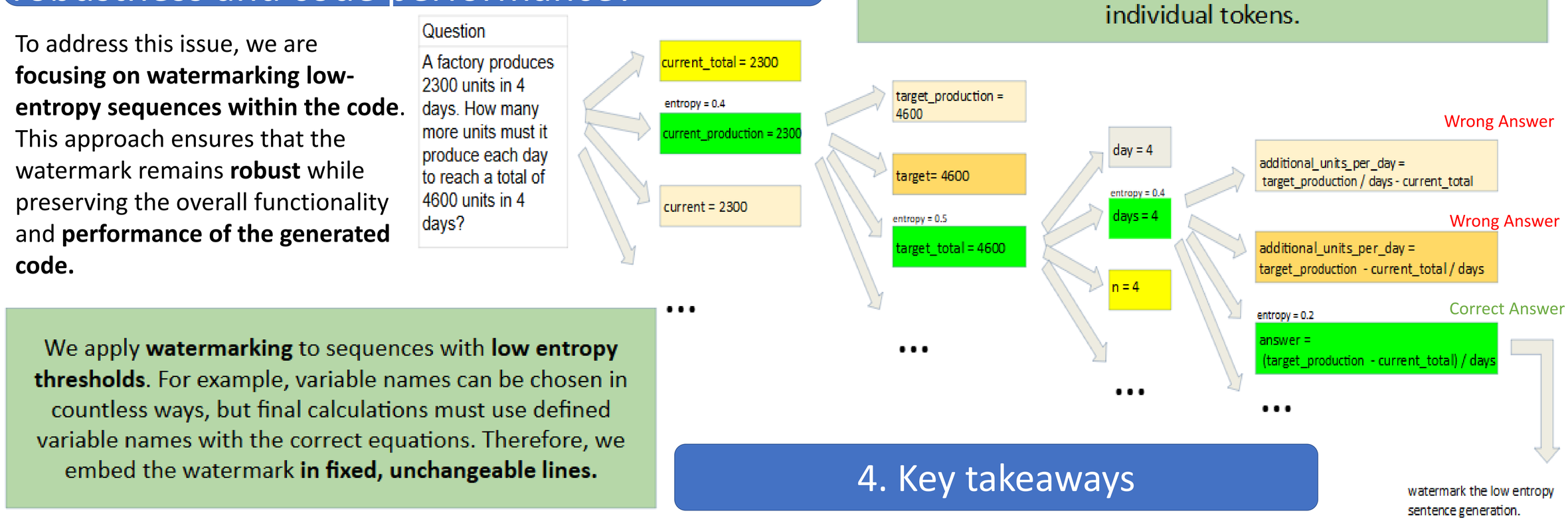



```
Generated Code:
def prime_check(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
```

3. How SLW improve the watermark robustness and code performance?

To address this issue, we are **focusing on watermarking low-entropy sequences within the code**. This approach ensures that the watermark remains **robust** while preserving the overall functionality and **performance of the generated code.**

The term "**sequence length**" refers to the beam search algorithm generating each lines of code rather than individual tokens.



We apply **watermarking** to sequences with **low entropy thresholds**. For example, variable names can be chosen in countless ways, but final calculations must use defined variable names with the correct equations. Therefore, we embed the watermark **in fixed, unchangeable lines.**

4. Key takeaways

Performance:

- Measure performance using Humaneval dataset.
- Ensure the generated code runs efficiently without significant overhead

Watermark Detection:

- Implement a watermark embedding method for code generation.
- Evaluate the effectiveness of watermark detection in various scenarios.

Analysis:

- Use open-source LLMs designed for code generation.
- Compare results across different models and configurations.

Goal:

- Achieve a balance between code performance and watermark robustness.

Method	Humaneval	
Watermark	passk	auroc
WLLM	29.6	0.402
SWEET	32.6	0.943
SLW	TBD	TBD

The work of J.H. was supported in part by College of Science and Mathematics Dean's Undergraduate Research Fellowship through fellowship support from **Oracle, project ID R20000000025727**.

NCRC 2025