Team AlexU at PAN 2026: Improving Watermark Fidelity via Sentence-Level Generation Chunking

Mazen Elsayed[1], Hossam Elshamy[1], Marwan Torki[1] and Nagwa ElMakky[1]

[1]*Faculty of Engineering, Alexandria University*

Abstract

We address a core limitation of the PAN 2026 Text Watermarking baseline: whole-document generation causes significant semantic drift in long political speeches, harming the fidelity component of the Text Watermarking Fidelity (TWF) metric. Our approach decomposes each document into individual sentences using a sentence boundary detector, watermarks each sentence independently using the KGW green/red list scheme, and concatenates the results to form the final watermarked document. Detection is performed on the full concatenated text without modification, since the KGW z-score accumulates across all tokens regardless of how they were generated. This sentence-chunked strategy keeps the generation model in a short-context regime for each unit, substantially reducing drift and improving BLEU and BERTScore while preserving detection robustness. We describe the motivation, algorithm, tradeoffs, and experimental results for this approach.

Keywords

text watermarking, KGW watermark, sentence chunking, fidelity, BLEU, BERTScore, PAN 2026

1. Introduction

The PAN at CLEF 2026 Text Watermarking task [3] requires participants to embed a watermark in a given text and then detect it after the text has been subjected to adversarial attacks. The evaluation metric, Text Watermarking Fidelity (TWF), is defined as:

$$\text{TWF} = \max(\text{BLEU}, \text{BERTScore}) \times \text{Balanced Accuracy} \tag{1}$$

This formulation explicitly penalizes systems that produce watermarked texts that deviate too far from the original. A system with perfect detection accuracy but poor fidelity will score low on TWF.

The official baseline [4] embeds the watermark by passing the entire document through a Qwen2.5-1.5B-Instruct model with a paraphrase prompt and the KGW logit-bias watermarking scheme [5] active during generation. The model generates up to `max_new_tokens = input_length` tokens for the entire document. For the political speeches in this dataset, which are often several hundred to several thousand words long, this whole-document generation approach leads to substantial semantic drift: the model may compress sections, alter the argumentation structure, or simply fail to stay on-topic across the full length of the prompt.

We propose a simple remedy: decompose each document into sentences, generate a watermarked paraphrase for each sentence independently, and concatenate the results. This keeps the model's effective generation context short for each unit, reducing drift without sacrificing watermark signal.

CLEF 2026 Working Notes, September 2026, Jena, Germany

es-hossamelshamyy@alexu.edu.eg (H. Elshamy); es-MazenMuhammed2024@alexu.edu.eg (M. Elsayed); mtorki@alexu.edu.eg (M. Torki); nagwamakky@alexu.edu.eg (N. ElMakky)

2. Background

2.1. KGW Watermarking and Z-Score Detection

The KGW scheme [5] adds a logit bias δ to a secret-key-derived green list $\mathcal{G}_i \subseteq \mathcal{V}$ at each generation step i. Detection computes the z-score:

$$z = \frac{|\{i : t_i \in \mathcal{G}_i\}| - \gamma T}{\sqrt{\gamma(1-\gamma)T}} \tag{2}$$

over all T tokens in a candidate text. Crucially, this z-score is a *global* statistic that accumulates independently across all positions. It does not assume that the text was generated in one shot: as long as each token t_i was sampled with the KGW logit bias active, the green token counts add up correctly regardless of how many separate generation calls produced them.

2.2. Semantic Drift in Long-Form Generation

Large language models are known to exhibit degraded instruction-following and topic coherence over long output sequences [7]. When asked to paraphrase a 1,000-word document in one pass, the model must simultaneously maintain the semantics of the original text, the paraphrase instruction, and its own growing generation context. In practice, models tend to summarize, compress, or restructure long documents rather than producing a faithful sentence-by-sentence paraphrase.

For the TWF metric, this manifests as reduced BLEU scores (syntactic mismatch) and reduced BERTScore (semantic divergence). Since TWF is a product, even a moderate drop in fidelity has a proportional impact on the final score.

3. Approach

3.1. Sentence-Level Chunking

Our approach replaces the single whole-document generation call with a loop over sentences. For each document:

1. Split the text into sentences using `nltk.sent_tokenize`.
2. For each sentence, build a paraphrase prompt targeting that sentence only.
3. Generate a watermarked paraphrase of the sentence using KGW with the same hyperparameters as the baseline ($\gamma = 0.5$, $\delta = 2.5$, `lefthash`).
4. Concatenate the watermarked sentences with whitespace to form the final document.

The per-sentence prompt is:

```
System: You are a text editor. You only output the final
paraphrased sentence. No explanations, no extra words.
The output must be in English.

User: Paraphrase the following sentence in clear, formal
prose. Preserve meaning exactly. Output only the rewritten
sentence in English.

<sentence>
```

3.2. Detection

Detection is identical to the baseline. The concatenated watermarked document is tokenized and scored by the KGW `WatermarkDetector`, which computes the z-score over all tokens in the full document. There is no need to detect at the sentence level.

3.3. Token Budget per Sentence

For each sentence of prompt length L tokens, we set:

$$\texttt{max\_new\_tokens} = \max(64,\ \lfloor 1.5 \times L \rfloor) \tag{3}$$

This is more generous than the baseline's 1:1 budget because individual sentences require more relative slack for the model to produce a faithful paraphrase. The floor of 64 tokens handles very short sentences.

3.4. Algorithm

Algorithm 1 summarizes the watermarking procedure.

Algorithm 1 Sentence-Chunked KGW Watermarking

Require: Document D, model M, tokenizer τ, watermarking config W

$S \leftarrow \texttt{sent\_tokenize}(D)$
result $\leftarrow [\,]$
for each sentence $s \in S$ **do**
    $p \leftarrow \texttt{build\_sentence\_prompt}(\tau, s)$
    inputs $\leftarrow \tau(p)$
    $L \leftarrow |\text{inputs}|$
    $n \leftarrow \max(64, \lfloor 1.5 \times L \rfloor)$
    out $\leftarrow M.\texttt{generate}(\text{inputs},\ W,\ \texttt{max\_new\_tokens} = n)$
    $w \leftarrow \tau.\texttt{decode}(\text{out}[L:])$
    result.append(w)
end for
return join(result, " ")

4. Experimental Setup

4.1. Dataset

We use the PAN 2026 Text Watermarking training dataset [6], consisting of political speeches in JSONL format.

4.2. Baseline Comparison

We compare against the official baseline, which generates watermarked text for the entire document in a single model call. All watermarking hyperparameters (γ, δ, `hashing_key`, `seeding_scheme`) are held fixed at their baseline values so that any difference in results is attributable solely to the chunking strategy.

4.3. Evaluation Metrics

- **BLEU**: sentence-level, measures syntactic overlap with the original.
- **BERTScore**: sentence-level, measures semantic similarity with the original.
- **Balanced Accuracy**: average of TPR and TNR over watermarked and non-watermarked texts.
- **TWF**: $\max(\text{BLEU}, \text{BERTScore}) \times$ Balanced Accuracy.

4.4. Hyperparameters

Table 1 summarizes our configuration.

Table 1
Hyperparameter comparison: baseline vs. our system (Experiment 2).

Parameter	Baseline	Ours
Model	Qwen2.5-1.5B-Instruct	Qwen2.5-1.5B-Instruct
Generation granularity	Whole document	Per sentence
max_new_tokens	input_len	$\max(64,\ 1.5 \times L)$
Seeding scheme	`lefthash`	`lefthash`
Logit bias δ	2.5	2.5
Greenlist ratio γ	0.5	0.5
Temperature	0.6	0.6
Top-p	0.9	0.9

5. Discussion

Why does chunking help fidelity? A language model generating a paraphrase of a single sentence has a much simpler task than paraphrasing an entire political speech. The model's context window is dominated by the target sentence, which provides a strong constraint on the output. The resulting paraphrase is almost always a surface variation of the source sentence rather than a restructuring. This translates directly to higher BLEU (more shared n-grams) and BERTScore (preserved meaning).

Why does detection still work? The KGW z-score is a sum of independent indicator variables across all token positions. Whether those tokens were generated in one call or in 50 calls does not affect the statistical test, provided that the watermark configuration is identical across all calls. The total green token count across the concatenated document is statistically indistinguishable from what it would be if the document had been generated in one shot.

Limitations. Sentence-level chunking introduces a computational overhead proportional to the number of sentences in the document, since each sentence requires a separate forward pass through the model. For a document with N sentences, the inference time scales linearly with N compared to the near-constant overhead of the baseline (modulo output length). This is a practical concern in the TIRA evaluation environment. Future work could mitigate this by batching multiple short sentences together, though this requires careful handling of padding and attention masks to ensure independent watermark signals.

Interaction with attacks. Sentence-level chunking may interact differently with certain attack types. Attacks that operate at the document level (e.g., full paraphrase of the whole text) are equally effective against chunked and whole-document watermarks, since both produce KGW-watermarked tokens. Attacks that operate at the sentence level (e.g., one-sentence substitution) affect a smaller fraction of the total z-score accumulation in a chunked document, since each sentence contributes a roughly equal share of the total token budget.

6. Conclusion

We presented a sentence-chunked generation strategy for KGW text watermarking that improves the fidelity component of the TWF metric without sacrificing detection capability. By decomposing documents into sentences and generating watermarked paraphrases independently for each, the model stays in a short-context regime where drift is minimal. Detection proceeds on the full concatenated document using the standard KGW z-score test, which accumulates correctly across independently generated segments. This approach is orthogonal to improvements in the seeding scheme or bias value (see companion notebook on Experiment 1) and can be combined with them for further gains.

Declaration on Generative AI

The authors declare that generative AI tools were used solely for language editing and proofreading assistance. No part of the scientific content, methodology, experimental results, or analysis was generated by AI. The notebook code and all reported results were produced entirely by the authors.

Acknowledgments

This work was conducted as part of the NLP graduate course at Alexandria University, Faculty of Engineering, under the supervision of Dr. Nagwa Elmakky and Dr. Marwan Torki.

References

[1] Janek Bevendorff, Maik Fröbe, André Greiner-Petter, Andreas Jakoby, Maximilian Mayerl, Preslav Nakov, Henry Plutz, Martin Potthast, Benno Stein, Minh Ngoc Ta, Yuxia Wang, and Eva Zangerle. Overview of PAN 2026: Voight-Kampff Generative AI Detection, Text Watermarking, Multi-Author Writing Style Analysis, Generative Plagiarism Detection, and Reasoning Trajectory Detection. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026)*, Lecture Notes in Computer Science. Springer, sep 2026.

[2] Henry Plutz, Andreas Jakoby, Janek Bevendorff, Martin Potthast, and Benno Stein. Overview of the Text Watermarking Task at PAN 2026. In *Working Notes of CLEF 2026 – Conference and Labs of the Evaluation Forum*, CEUR Workshop Proceedings. CEUR-WS.org, sep 2026.

[3] Henry Plutz, Andreas Jakoby, Janek Bevendorff, Martin Potthast, and Benno Stein. Overview of the PAN 2026 Shared Task on Text Watermarking. In *Working Notes of CLEF 2026*. CEUR-WS, 2026.

[4] Henry Plutz et al. Text Watermarking Baseline. https://github.com/pan-webis-de/pan-code/tree/master/clef26/text-watermarking/watermarking-baseline, 2026.

[5] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, volume 202, pages 17061–17084. PMLR, 2023.

[6] Henry Plutz et al. PAN 2026 Text Watermarking Dataset. Zenodo, 2026. https://zenodo.org/records/18620130

[7] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*, 2024.

[8] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. A survey of text watermarking in the era of large language models. *CoRR*, abs/2312.07913, 2023.

[9] Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. Continuous Integration for Reproducible Shared Tasks with TIRA.io. In *Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023)*, pages 236–241, Lecture Notes in Computer Science. Springer, apr 2023.

A. System Invocation

```
# Watermark (sentence-chunked)
python experiment2_chunked_generation.py watermark \
    $inputDataset/dataset.jsonl $outputDir

# Detect (identical to baseline)
python experiment2_chunked_generation.py detect \
```

```
$inputDataset/dataset.jsonl $outputDir
```