ZPJa: Watermarking Technical Report

Antonín Jarolím
xjarol06@fit.vut.cz

Vojtěch Eichler
xeichl01@fit.vut.cz

Abstract

In this work, we compare and investigate the behavior of two watermarking methods, RedGreen and GumbelSoft watermarks, for large language models (LLMs). Rather than focusing on the detectability of the watermark themselves or their weaknesses in regard to attacks against them, we focus on different, quite practical tasks. We mainly investigate how detection works when presented with different keys (then those used for generation) and how different models with the same key behave when it comes to detection.We verified that detecting a watermark with a certain key won't recognize the watermark if the text was watermarked with a different key. We observed that when different models share the same tokenizer, detection can be performed only once, reducing the need for resources. We also investigate the influence of text entropy and its length on detectability, confirming the positive correlation of both metrics.

1 Introduction

Since the release of ChatGPT[1] and similar chat interfaces for LLMs, the presence of machine-generated text has skyrocketed. There are many scenarios, such as academia, where it would be very useful to be able to tell, whether a LLM generated the text or a human. This issue comes up even in the question of protecting ownership or monetary exploitation of such models. The companies providing text-generation services could also use this information not to train their new models on their generated text. Again, being able to confidently say that a text was generated by a concrete model can be very useful.

Many watermarking methods have emerged but aren't commonly used (at least publicly) and have only been around briefly. We therefore investigate some of the intricacies of two of the methods, RedGreen and GumbelSoft watermark.

[1] chatgpt.com

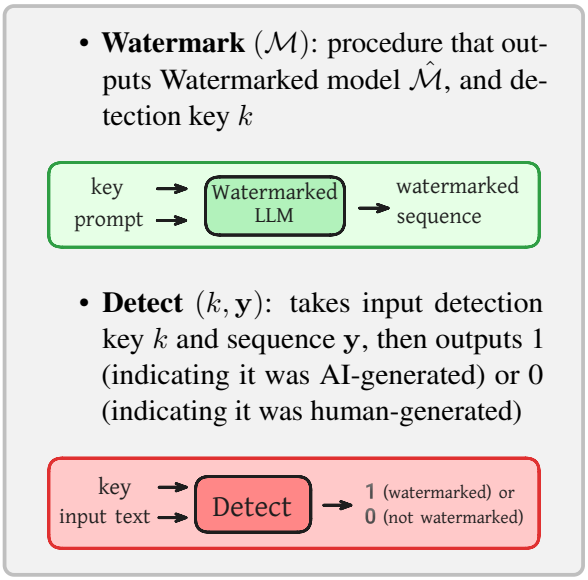

Figure 1: Illustration of the watermarking and detection procedures.

Therefore, we designed a range of experiments that lead to the following key contributions:

1. We demonstrate the detectability of watermarked text across various scenarios.
2. We show that models within the same family, sharing a tokenizer, can effectively use the same watermarking key.
3. We establish and analyze the relationship between entropy, generated text length, and detectability, confirming it behaves as anticipated.

2 Task Definition

The main task is to apply both watermarks to different models and different datasets. Both watermarks are built as a two-component system, with the first component being a *generator*, which generates text token by token but also applies the watermark to

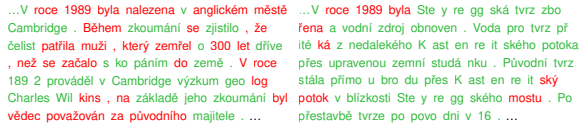

Figure 2: Visualization of red and green tokens for unwatermarked (left) and watermarked (right) texts. We can observe, that the right texts consits of more green tokens, which is expected, as Red-Green watermarking method boosts the logits of green tokens.

each of the tokens. The second component, *detector*, detects whether a text contains the watermark given a text and a key. The scheme can be seen in Figure 1.

The task is to reimplement these methods and create an interface that allows changing models and experimentation. Run generation and detection on multiple datasets using multiple models as described in Section 4.

3 Method

This section covers the watermarking methods we selected. We chose Red-Green and GumbelSoft watermark. Red-Green is very basic and easy to understand watermark, which we chose as a baseline due to the simplicity. GumbelSoft is more complex and more robust watermark, which we chose as we believe it's at least very close to SOTA (Fu et al., 2024).

3.1 Red-Green watermark

Red-Green (Kirchenbauer et al., 2024) is a very simple watermarking method and we used the simplest version, which uses global red and green lists. These lists are the foundation of Red-Green watermark. According to a watermark key, the vocabulary is stochastically split into two mutually exclusive lists. During generation, each step logits are altered by adding a small constant to logits for those tokens that are on a green list. This favors green list tokens and makes them appear more often than without the watermark, as illustrated in Figure 2.

Detection is quite intuitive as well. The detector, based on the key, will generate a red and green list as well. Now, assuming the key during detection is the same as during generation, the lists will be exactly the same. This fact allows the detector to calculate statistics, whether the words that he assumes should be favored are actually favored in the provided text. The statistic used to decide whether a text is watermarked is a z-score in this form:

$$z = \frac{(|s|_G - \gamma T)}{\sqrt{T\gamma(1-\gamma)}} \quad (1)$$

where $|S|_G$ is the number of tokens that belong to the green list from the text in detection, γ is a green list size ($\gamma \in [0, 1]$), and T is the length of the text in tokens. Based on the z-score value, the decision can be made by thresholding by some constant value, and when the z-score reaches or surpasses the value, the text is considered watermarked. In other words, the null hypothesis is, that the text is not watermarked, based on the z-score, we reject the null hypothesis once z-score reaches some thresholding value.

3.2 GumbelSoft

GumbelSoft watermark (Fu et al., 2024) is much more complex, but the idea of favoring some tokens over others and detecting this bias is still present.

As with Red-Green, we generate token by token and alter the logits. The Watermark key is used as a seed for random number generators. In each step, the generator samples a Gumbel-distributed vector based on the context (n-gram, in our work we always use 3-gram, meaning that 3 preceding tokens are used as a context). This vector is the size of the vocabulary and is then added to the logits vector. Once the logits are altered, the generator treats these logits as the output of the model and samples the next token. This process is repeated for each step. The pseudocode for the generation can be seen in figure 3.

Input: prompt r, LLM $\mathcal{M}$, temperature τ
Output: Watermarked sequence $w_1, \ldots, w_T$
1: **for** $t = 1, \ldots, T$ **do**
2: Logits $l_t \leftarrow \mathcal{M}(r, w_{1,\ldots,t-1})$
3: Watermark key $\xi_t \leftarrow$ hash context to a Gumbel-distributed vector
4: $w_t \leftarrow$ sample from softmax$((\xi_t + l_t)/\tau)$
5: **end for**
6: **return** $[w_1, \ldots, w_T]$

Figure 3: Pseudocode for GumbelSoft generator.

During detection, the null hypothesis is that the text is not watermarked. We can again calculate a z-score, which can help us reject the null hypothesis;

the formula is as follows:

$$S = \frac{\sqrt{6T}}{\pi}\left(\frac{\sum_{i=1}^{T} s_i}{T} - \gamma\right) \quad (2)$$

where T denotes the length of the text, γ denotes Euler-Mascheroni constant and s_i is a per-token score. The per-token score is one element from a Gumbel-distributed vector. This vector is created the same way as during generation, that is, by hashing the context (n-gram) and sampling Gumbel distribution. Again, the vector is the size of the vocabulary, and the per-token score for the token is the element of the vector corresponding to the token position in the vocabulary (Fu et al., 2024).

4 Experimental Setup

Since one of the primary tasks involves monitoring entropy, we deliberately selected datasets with varying entropy levels. As a representative low-entropy dataset, we chose SQuAD (Rajpurkar et al., 2016) and its Czech translated version. This dataset comprises questions created by crowd workers based on Wikipedia articles, alongside corresponding answers.

For our purposes, we utilize only the questions from this dataset as prompts for language generation. The answers, which are spans extracted from the Wikipedia articles, are typically concise and factual. This characteristic supports our classification of SQuAD prompts as low-entropy, given the limited variability and factual basis of the correct answers.

For the high-entropy datasets, we selected OpenGen (Krishna et al., 2023) and CNC_Capek (Čermák et al., 2007) datasets for English and Czech experiments, respectively. OpenGen comprises stories and dialogues, offering prompts with significant variability and complexity. Similarly, the CNC_Capek dataset utilizes random passages from Karel Čapek's works, further contributing to the high-entropy nature of the prompts.

We randomly sampled 7711 prompts from each of these datasets. For all of our experiments, we used two machines with NVIDIA GeForce RTX 3090 for text generation. The maximum number of generated tokens was set to 700; however, the final length of the texts varies, as the model may generate an <EOS> token, at which point the generation process is terminated. For the decoding parameters, we use top-p=0.9 and temperature=1.2.

The z-score can be thresholded to derive detection metrics. Therefore, it would be feasible to directly evaluate detectability using metrics such as accuracy or a confusion matrix. We instead use the z-score as our primary evaluation metric (and refer to it as detectability through the paper), because of the significant difference in z-score between watermarked and unwatermarked text in our datasets. With an appropriately chosen threshold, the accuracy would reach 100%, making it less informative for nuanced analysis.

The following is a list of experiments designed to demonstrate the functionality of watermarking and the ability to distinguish watermarked text.

4.1 Detection with different keys

A primary concern in watermarking is the management of keys. In our experiments, we generate a variety of texts using different keys and employ all keys for detection across texts generated with other keys. We demonstrate that detection is accurate only when the key used for both generation and detection is the same. Used models for generation were *meta-llama-Llama-3.1-8B* for english and *csmpt7b* (Fajčík et al., 2024) model for czech prompts.

4.2 Detection with different models

In this experiment, we employ various different models to generate watermarked texts from the same prompts from OpenGen dataset with the same key. More concretely, we use *meta-llama/Llama-3.1-8B*, *llama-Llama-3.2-3B* (Meta, 2024), *Ministral-8B-Instruct-2410* and *Mistral-7B-v0.3* (Jiang et al., 2023) models. Notably, while the tokenizer differs between versions of the LLaMA models, it remains consistent across the smaller and larger versions within the Mistral model family.

4.3 Exploring the effects of length and entropy on detectability

We generate texts from OpenGen using *llama-Llama-3.1-8B* and compute the number of generated tokens and entropy of the generated text using equation:

$$H(X) = -\frac{1}{N}\sum_{t=1}^{N} \log P(x_t \mid x_{<t}) \quad (3)$$

We assume that the higher the entropy, the better it is for watermarking. In a situation where the

generated text has very low entropy, it should be hard for the watermark to make an impact. Imagine a prompt "*Generate the word foo three times with a space between*". Reasonably good model should produce the sequence "*foo foo foo*" and there's no reason for model to generate anything else, we prefer this result, but how could the watermark have an impact?

Similarly, we expect the watermark to have more impact on longer texts as there are more tokens. More tokens mean more opportunities to skew the resulting token and actually watermark effectively. One could imagine a similar extreme case as for the entropy, that generating one watermarked token is not really sufficient for the statistics of watermark detectors to work effectively.

5 Results and Analysis

In this section, we present our findings and analysis of our results. These experiments are described in detail in the Section 4. This section will go through all three of the tasks.

5.1 Detectability with different keys

For each combination of model and dataset of model-corresponding language, we generate data using 5 different keys with Gumbel watermark. Then, for each set of generated texts, we run the detection algorithm using all 5 keys. The main goal of this task is to confirm, that the z-score for the detection with the corresponding key is high, while the z-score using different key is low. We can see the distribution of z-scores for the Czech high-entropy dataset - CNC_CAPEK, Gumbel watermark in the Figure 4. The Gumbel noise described in Section 3 for a detection with unmatched key is completely different from the noise generated with the correct key (the one we generated data with). The noise generated with different key is essentially independent of selected tokens, therefore, detection algorithm cannot succeed with different key.

We can see, that the distributions matches our expectations. Distribution for not matching keys is the same as the distribution for unwatermarked text. We can also notice, that if we would draw a horizontal line at value around z=5, we would separate correctly detected watermarked texts from correctly detected unwatermarked text. Therefore, the accuracy of detection is 1, when outliers are not taken into account.

Throughout our experiments, we did not observe any significant change in the distribution of z-scores, therefore, we show results for other combinations of datasets and models in Appendix A. We also repeated this experiment for Red-Green watermark and observed the same results.

5.2 Detectability with different models

In this task, we fix the key for all models and generate text from all of them. Essentially, detection algorithms described in Section 3 do not depend on model weights, because they just check the correlation of the noise and the generated text. On the other hand, generated noise corresponds to the vocabulary, i.e., for each word in a vocabulary, we generate one noise value. If the correlation of high noise value and selected tokens in the entire text is high, then the z-score given by detection algorithm is also high. By shuffling the vocabulary ids, we make the noise independent of them, making it impossible for the algorithm to succeed detection. Similarly, when the vocabulary differs (in the even more extreme case even in the number of tokens), the correlations is lost completely.

We hypothesize, that we could try to heuristically match tokens from different tokenizers, using e.g. Levenshtein distance, to find at least some correlation. However, the detection might fail for many cases, especially for short generated texts. On the other hand, for some long works such as university theses, it could be sufficient to proof correlation. Further evaluation and experiments would be needed to support this hypothesis.

The experiment using OpenGen data can be seen in the Figure 5. We can see, that the detection distributions are the same for models with the same tokenizers and that the distribution differs for models with different tokenizers. We can also observe, that the z-score is very low, when the tokenizer is different, as it corresponds to no correlation and therefore has similar distribution as unmatched generation-detection key pair in previous section. The experiment using red-green watermark ended up similar, and therefore is discussed in Appendix B.

The fact, that the distribution is the same, when the same tokenizer is used, can have practical consequences. For example, if a company offers text generation services with different models, they can use the same key and tokenizer for all of them. Then, at detection time, then can save resources by running detection only once for all models. If the

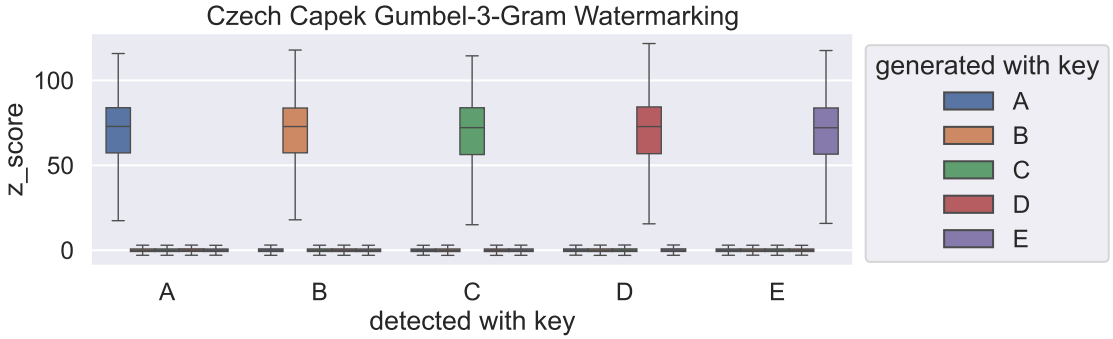

Figure 4: The distribution of z-scores across texts generated with keys {A, B, C, D, E} and detected with all other keys shows that high z-scores are observed only when the generated and detected keys match. In contrast, when there is a mismatch between the generated and detected keys, the z-scores are consistently low, approaching zero. Note that the values used for keys are only placeholders for a large numeric sequences used in the real implementation.

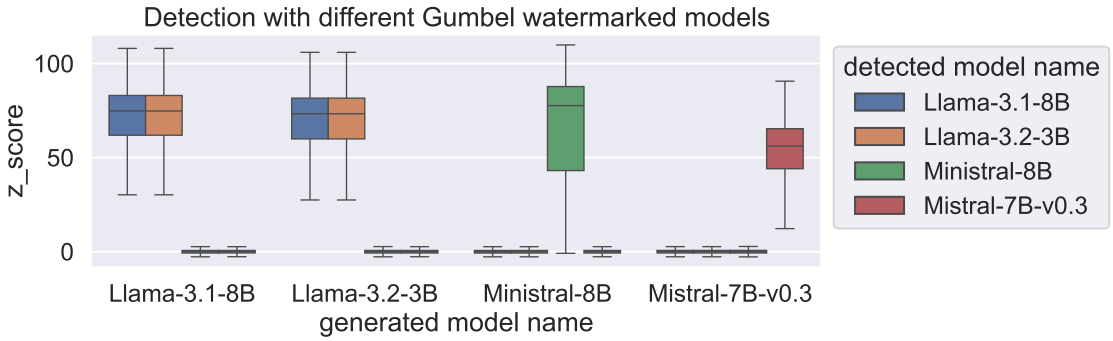

Figure 5: The distribution of z-scores across different models using the same key. The z-score distributions are the same for llama models, because the tokenizer is the same. On the other hand, when the tokenizer differs, the detection is different. That can be seen for models from Mistral family.

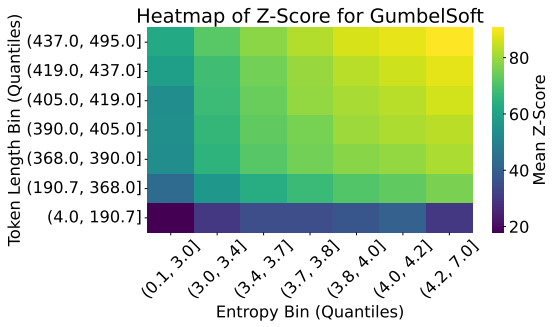

Figure 6: Effect on entropy and length of generated text to the detectability with Gumbel watermark.

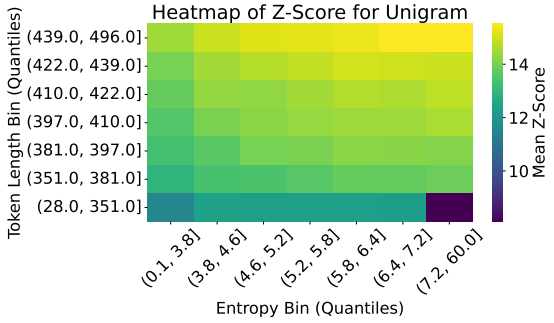

Figure 7: Effect on entropy and length of generated text to the detectability with Unigram watermark.

tokenizer or keys would differ, they would need to run detection for each pair.

5.3 Entropy and text length effect on detectability

In this task we wanted to understand the effect of entropy and text length, with our hypothesis being that higher entropy and longer texts will increase detectability. Therefore, we have generated text from OpenGen dataset, using *llama-3.1-8B* model, we have noted the length of generated text in tokens and we have computed the entropy of the text using Eq 3, given the model. Then, we used quantile binning, to divide data into equally sized groups of texts. For each group, we have computed the mean of the z-score, comprising the detectability of the group.

As we can see in the heatmaps in Figures [6, 7] for both GumbelSoft and Red-Green watermark, our hypothesis can be supported. More precisely, we can see, that there is a positive correlation between the z-score and the length of the text. This is expected, as we gather more observations to compute the statistics with. There is also a positive correlation between z-score and entropy, which is also expected, as the watermarking technique does not have much strength for a low entropy prompts. I.e., it is almost certain how the generated text should look like for a low entropy text, therefore, the probabilities for these expected tokens will be high and low for other. By adding the Gumbel noise, or increasing the logits of green tokens, we will still not change the distribution much, therefore the z-score will be low.

We could also speculate, that the entropy has higher impact on Gumbel watermark, given that the z-score (heat) varies more in the x-dimension than for Red-Green watermark. However, more experiments or evaluations would be needed to support this hypothesis.

6 Conclusion

In this work, we verified that detection will only detect watermarks when the generation process uses the same key as the detection, and different keys are not detected and behave as unwatermarked.

In the next set of experiments, we used the same watermark keys for different models. We were curious how the detection and z-scores would vary, given the fact that the detection doesn't depend on the model weights. We found out that models from the same family with the same tokenizer will be indistinguishable to the detection.

We also investigated whether our hypothesis that higher entropy and longer texts increase detectability. We confirmed this hypothesis with the last set of experiments, where we calculated entropy for all generated texts, calculated their lengths, ran detection, and visualized all this data.

References

Martin Fajčík, Martin Dočekal, Jan Doležal, Karel Beneš, and Michal Hradiš. 2024. Benczechmark: Machine language understanding benchmark for czech language.

Jiayi Fu, Xuandong Zhao, Ruihan Yang, Yuansen Zhang, Jiangjie Chen, and Yanghua Xiao. 2024. Gumbelsoft: Diversified language model watermarking via the gumbelmax-trick.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2024. A watermark for large language models.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense.

Meta. 2024. The llama 3 herd of models.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

F. Čermák et al. 2007. Capek: Korpus pouze vlastních textů karla Čapka. Ústav Českého národního korpusu FF UK, Praha. Released corpus.

A Further analysis of different keys experiment

We thought that higher-lower entropy of datasets will have some impact on the distribution of z-scores for a correct detection. This hypothesis has not been confirmed by our experiments. As we can see, the distribution varies slightly, e.g., we can see the distribution of z-scores for czech squad in Figure 8. There is higher variance compared to Figure 4, which matches our hypothesis, that the detectability for higher-entropy datasets would be lower. However, when we repeat the same experiment with different datasets with english data, we cannot make the same observation. The Figure 9 and Figure 10 show z-score distribution for OpenGen and Squad dataset, respectively. The opposite conclusion (higher variance for high-entropy dataset) can be seen here.

Note, that even though the variance is higher, we can still find a threshold which would have close-to-perfect accuracy. We also took a look at the generated texts, which had z-score smaller then 1, even with a correct generated-detected key pair. We can observe text competitions in the Figure 11. The firs observations is, that all of them are very short. Second observation is, that the texts have only a very low entropy, e.g. in the extreme case, counts from 4 to 13. As discussed in Section 5.3, the detectability of such texts is expected to be low. Also, we do not really care about the detection result of such texts, and therefore we can conclude, that it does not matter, that the z-score is low on this texts.

B Different models with Red-Green watermark

Experiment results of generating data with the same keys for different models for Red-Green watermark can be seen in Figure 12. We can see, the the experiment results are the same, with a slightly varying z-score distribution, as the watermarking technique differs.

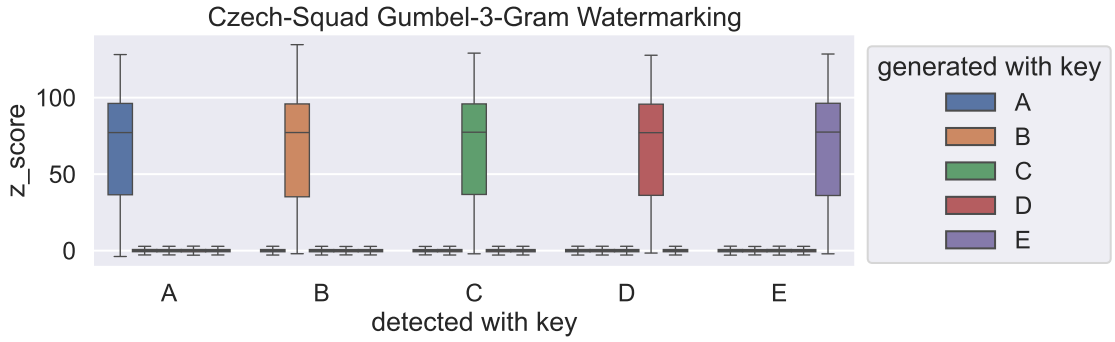

Figure 8: Distribution of z-scores for Squad translated to Czech language, *csmpt7b* model, 3-gram Gumbel watermark.

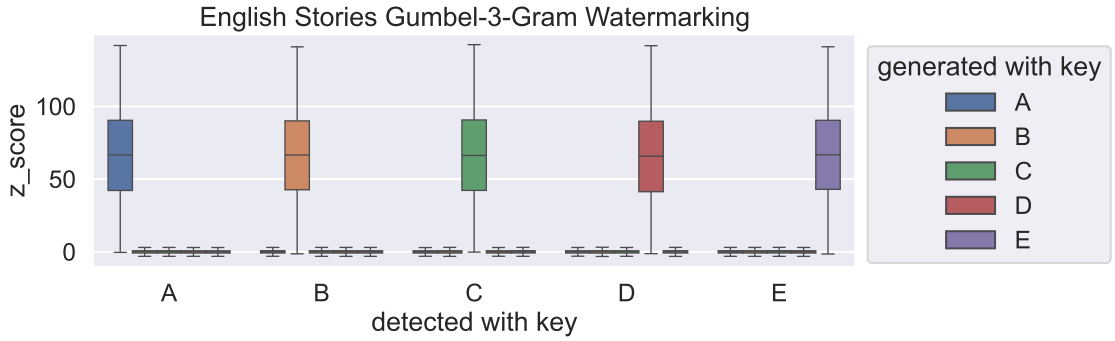

Figure 9: Distribution of z-scores for OpenGen data, *llama-Llama-3.1-8B* model, 3-gram Gumbel watermark.

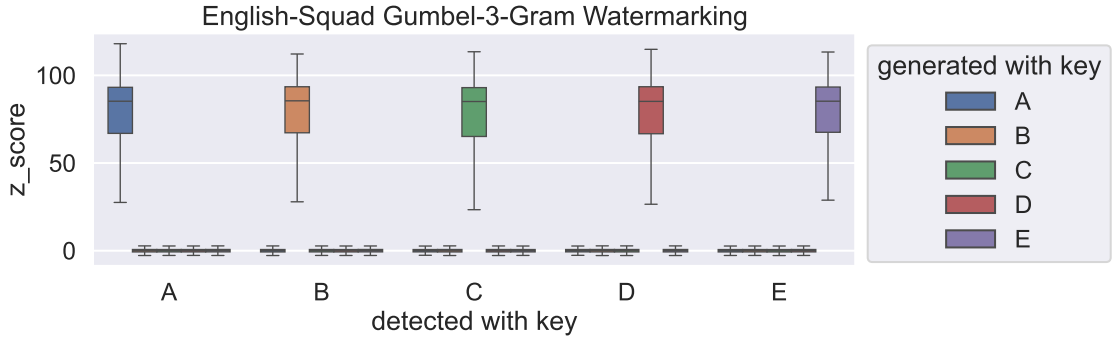

Figure 10: Distribution of z-scores for Squad data, *llama-Llama-3.1-8B* model, 3-gram Gumbel watermark.

```
her to the north.
miles too far to the north.
range.
2012.....  4 5 6 7 8 9 10 11 12 13
work-in no.  2.
divided by entrances.
al Park was declared a World Heritage Site by UNESCO in
1979.
tikal National Park was declared a UNESCO World
Heritage Site in 1979.
new action as well.
fine with 59 laps to go.
```

Figure 11: Each line consists of generation competition at which the z-score was very low, even for a good generated-detected key pair. The detection would fail on this example even with the best-tuned threshold.

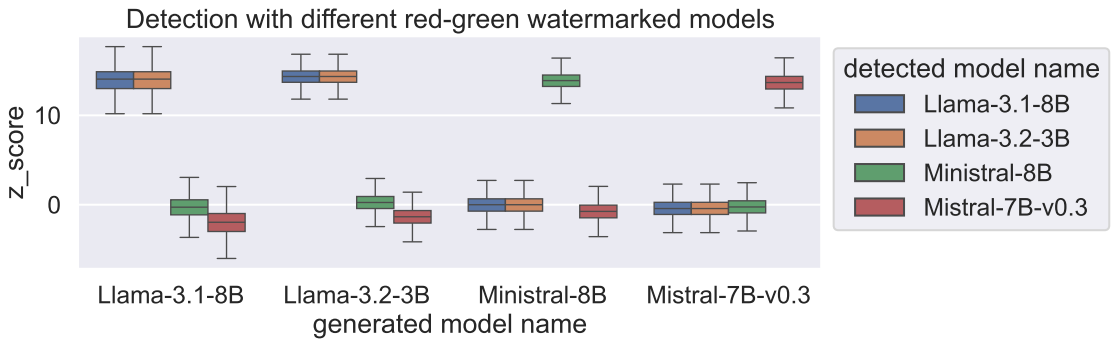

Figure 12: The distribution of z-scores across different models using the same key. The z-score distributions are the same for llama models, because the tokenizer is the same. On the other hand, when the tokenizer differs, the detection is different. That can be seen for models from Mistral family.