Evaluating LLMs' capability on Satisfying Lexical Constraint

Anonymous ACL submission

Abstract

Lexical Constrained Generation (LCG) is a fundamental task in text generation. Recent advancement of large pretrained language models (LLMs) has enabled prompt-based controlling for LCG. Despite growing interest in assessing LLMs' capabilities in various aspects, there remains a lack of thorough investigation. To address this gap, we systematically analyze the performance of LLMs on satisfying lexical constraints with prompt-based controlling, as well as their efficacy in downstream applications (such as recipe generation, table-to-text, profile writing, etc). Through extensive experimentation, we identified several key observations that elucidate the limitations of LLMs in LCG, including (1) position bias, where LLMs tend to satisfy constraints that appear in specific positions within the input; (2) insensitive decoding parameters, which minimally impact the performance of LLMs; and (3) the inherent complexity of certain constraints (i.e. compound word). We conclude that there is a complexity bottleneck: **LLMs still face significant challenges in consistently satisfying lexical constraints**. Additionally, we introduce the Divide and Conquer Generation strategy, effective for both white-box and black-box LLMs, significantly enhancing their performance in LCG tasks. This strategy boosts LLMs' success rate by 93% in the most challenging LCG task, which is 40% more than the baseline. Our analysis aims to provide valuable insights into the performance of LLMs in LCG, and our proposed strategy offers a pathway to more sophisticated and customized text generation applications.

1 Introduction

Lexical Constrained Generation (LCG) is a crucial task of text generation (Zhang et al., 2023a). By enforcing the inclusion of pre-specified words in the output, LCG facilitates the generation of more faithful and relevant texts. It is helpful for various

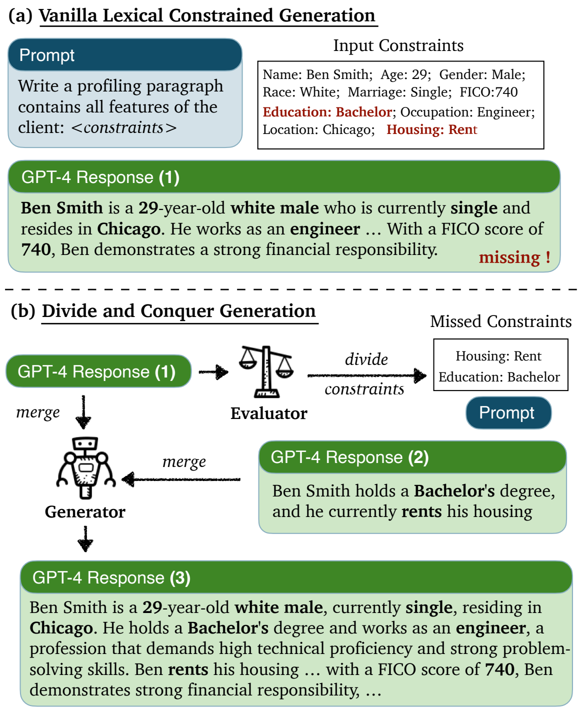


Figure 1: Sub-figure (a) illustrates that modern LLMs struggle to consistently meet complex lexical constraints in real-world scenarios. As shown in sub-figure (b), the Divide-and-Conquer Generation strategy divides the constraints into two parts (satisfied and missed), then generates response that with the missed constraints - i.e. response (2) - and merge them with the satisfied ones, enhancing the LLMs' ability to meet all specified constraints

real-world applications, such as dialogue generation (Knowles and Koehn, 2016), table-to-text generation (Chen et al., 2023), and recipe generation (H. Lee et al., 2020).

To generating text that adheres to lexical constraint effectively, previous works either design constrained decoding strategies, develop specialized models structure, or present refined mechanism (Sha, 2020; Lu et al., 2021; Qian et al., 2022; Meng et al., 2022). However, these approaches often come with significant drawbacks, such as high inference times, complex implementations, and unstable text quality. The recent advancements in pretrained large language models (LLMs) have showcased their robust few-shot capabilities (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023). Instruction tuning (Zhang et al., 2023b) has further enhanced LLMs' ability to generate text that meets controllable constraints as desired by humans. These developments make prompt-based controlling an increasingly efficient and practical method on tackling LCG task(Yang et al., 2022). Notably, prompt-based controlling has shown markedly superior strength and robustness compared to earlier methods for LCG (Sun et al., 2023; Ashok and Poczos, 2024), which motivate us to ask : *With prompt-based controlling, can LLMs consistently satisfy lexical constraints when generating text?*

Many recent works investigate in prompt-based controlling of LLMs (Sun et al., 2023; Zhang et al., 2023a; Ashok and Poczos, 2024). They conclude that LLMs shown effectiveness in satisfying lexical constraints. However, their experiments have typically involved relatively simple tasks with a narrow scope. This leaves a significant gap in detailed understanding of their proficiency and limitations when it comes to satisfying lexical constraints, and effectiveness in real-world applications.

To address this gap, we present a systematic analysis of the performance of LLMs in generating text under lexical constraints, and we also evaluate their utility in downstream applications where adhering to specific lexicons is crucial. Through extensive experiments, we conclude that **LLMs struggle to adapt to increasingly complex lexical constraints**. There is a complexity bottleneck: As the number of keywords increases, LLMs' performance decreases dramatically. We also observed that:

1. Position Bias: The position of each constraint within the prompt can substantially influence the model's output.

2. Insensitive Decoding Parameter: Decoding parameters are not highly sensitive for LLMs in LCG task, especially for temperature and top-k.

3. Inherent Complexity of compound words as constraints: LLMs tends to break down compound words into sub words, which can lead to misinterpretations or alteration of the intended meaning of the output significantly.

Additionally, we introduce an effective strategy - Divide and Conquer Generation - to enhance the ability of models to meet lexical constraints, which significantly improves performance, and helps LLMs achieve more satisfying results in downstream applications. Notably, the Divide and Conquer Generation strategy enables LLaMA-7b to improve the success rate by 93% in the most challenging LCG task, which is about 40% more over the baseline strategy. Our strategy is well-suited for both white-box and black-box models, making it an invaluable tool for a broad scope of application across diverse modeling environments.

Overall, our research conduct in-depth analysis on LLMs in satisfying lexical constraints, identify the current challenges faced by LLMs in satisfying lexical constraints, and provides a viable solution to these challenges, pave the way for more sophisticated downstream applications.

2 Lexical-constrained Generation

2.1 Task Setup

Following previous works(Lin et al., 2019; Zhou et al., 2023), we refer to constraints that require the generated text to include certain keywords in the output as lexical constraints. We consider an input prompt composed of a series of tokens, containing a set of constraints $X = [x_1, \dots, x_m]$, where x_i represent a keyword that must be included. The target output is a coherent sentence $Y = [y_1, \dots, y_N]$, with each y_i is a token. The task is to map the constraint set X into an appropriate sentence Y that both adheres to the prompt's requirements (e.g. generate a recipe) and satisfied the defined constraints(e.g. generate sentence that contain all given keywords) .

Evaluation Metrics We introduce two evaluation metrics in this study:

1. **Instance Success Rate** (R_{instance}): This metric evaluates whether each generated instance satisfies all specified constraints. It is defined as:

$$R_{\text{instance}}(X, Y) = \begin{cases} 1 & \text{if } X \subseteq Y, \\ 0 & \text{otherwise.} \end{cases}$$

2. **Keyword Coverage Rate** (S_{keyword}): This metric measures the proportion of input con-

straints included in the generated texts. It is calculated as:

$$R_{\text{keyword}} = \frac{\text{Number of Satisfied constraints}}{\text{Total number of constraints}}$$

Evaluate with LLMs We have conducted tests using various language models, including LLaMA2-7b-chat, LLaMA2-13b-chat, LLaMA3-8b-chat, GPT-3.5, and GPT-4. In these experiments, we tasked the models with generating outputs based on specific constraints. Unless stated otherwise, all experiments in this section utilized a greedy decoding strategy for generating responses. Prompt used in evaluation is attached to Appendix A.

2.2 Simple Constraints

We initiate our investigation with simple constraints, employing the CommonGen benchmark (Lin et al., 2019) to assess how well LLMs generate coherent sentences from a given set of concepts.

Experiment Setting. CommonGen (Lin et al., 2019) is a constrained commonsense generation task with lexical constraints. In this experiment, we treat each concept list in CommonGen as input constraints for LLMs to generate a proper sentence. We employ the instance success rate as the evaluation metric.

Evaluation Result. Figure 2 presents the results of experiments.GPT-3.5 and GPT-4 demonstrate impressive performance, achieving average instance success rates of 91% and 95% respectively across three distinct groups of instances. Conversely, LLaMA3-8b shows a less satisfactory average with a 63% coverage rate, while LLaMA2-13b achieves only a 55% rate. LLaMA2-7b records the lowest instance coverage among the evaluated models. This result suggests that the model's size significantly influences its ability to generate text that adheres to specified lexical constraints. Interestingly, LLaMA3-8b outperforms LLaMA2-13b, indicating that factors other than sheer model size may contribute to differences in model effectiveness.

2.3 Challenging Constraints

To increase the complexity of the constraints, we expanded the number of concepts that need to be incorporated into the generated text.

Experiment Setting. In this experiment, we randomly select concepts from the entire set of concepts within the CommonGen dataset to create a new, more challenging dataset. Then we repeat previous experiment setting to explore how well do LLMs adapt to increasingly complex constraints.

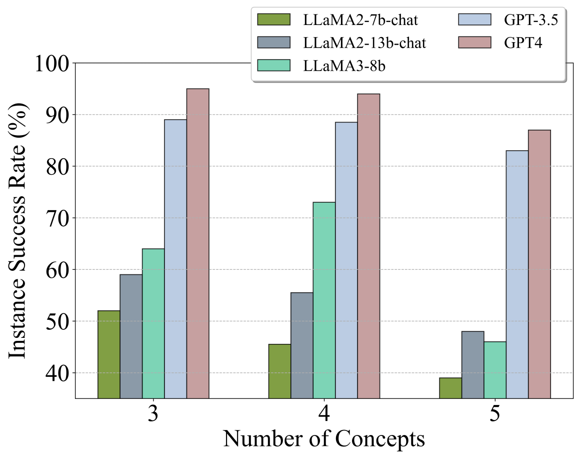


Figure 2: Experiment results on instance success rate by number of concepts.

Evaluation Result. As shown in Figure 3, there is a clear trend across all models, where the instance success rate declines as the complexity of constraints (i.e. number of concepts) increases. GPT-4 demonstrates slightly better resilience against rising complexity, maintaining a relative higher coverage rate across various groups of instances than other models. In contrast, as the number of concepts reaches 15, the performance of other models drops significantly. Notably, GPT-3.5 shows a significant decline in coverage rates; it drops from 98% to 13% as the number of concepts increases from 3 to 15. This sharp decrease eventually brings its performance in line with that of smaller models, such as LLaMA2-7b-chat and LLaMA2-13b-chat.

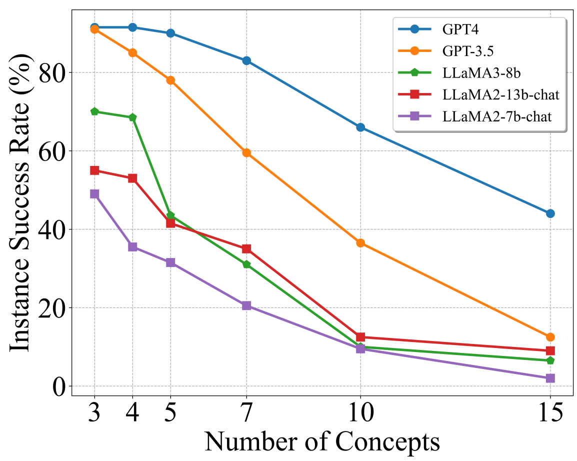


Figure 3: Experiment results on instance success rate by number of keywords.

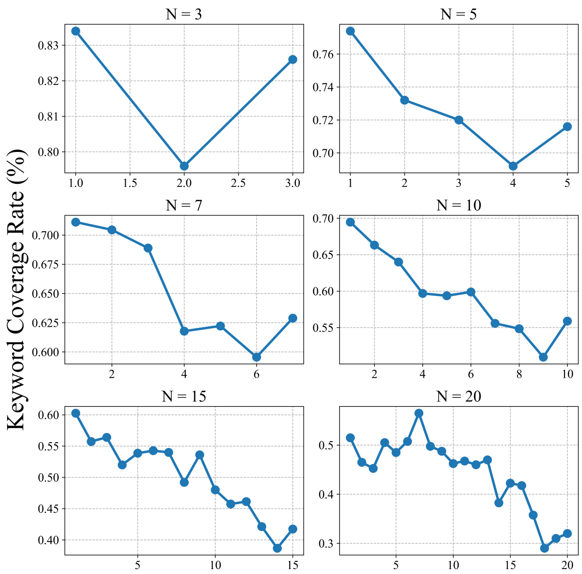


Figure 4: Experiment result on the position sensitive of LLaMA3-8b, presenting in terms of the keyword coverage rate (y-axis) for constraints placed at different positions (x-axis)

3 Sensitive Analysis

To better understand the factors causing LLMs to struggle with satisfying lexical constraints, we conducted a sensitivity analysis to investigate from various perspectives.

3.1 Position Bias

The constraints are placed at varying positions within the prompt. For example, consider the prompt:

> *Generate a sentence with the following keywords: mountain, cat, play, jump.*

Here, *mountain, cat, play, jump* serve as constraints. The word "mountain" is positioned earliest in the sequence, while the word "jump" appears at the end. Previous work finds (Wang and etc., 2023) in natural language understanding tasks, wherein it tends to select labels placed at earlier positions as the answer. We aim to investigate the position bias of LLMs in LCG task.

Experiments Setting We conduct experiments for 6 setting (number of keywords = [3,5,7,10,15,20]). For each setting with different specified number of keywords, we randomly select 100 sets of keywords, shuffle their positions, and conduct the experiment 20 times to ensure robustness. We evaluate the average keyword coverage rate for constraint in each position.

Experiment Result Our findings confirm that all LLMs exhibit a position bias, where keywords placed at different positions in the sequence lead to varying coverage rates. This bias is primarily attributed to either the primacy or recency effect, depending on the model. Some models, such as GPT-3.5, GPT-4, and LLaMA2-13b, are more influenced by the primacy effect, where keywords in earlier positions are more likely to be covered. Conversely, models like LLaMA2-7b and LLaMA3-8b demonstrate a stronger recency effect, prioritizing the most recently presented items. For instance, as illustrated in Figure 4, the keyword coverage rate decreases as the position increases from the first to the last. Keywords placed earlier in the input sequence (i.e., the prompt) are more likely to be covered than those in later positions.

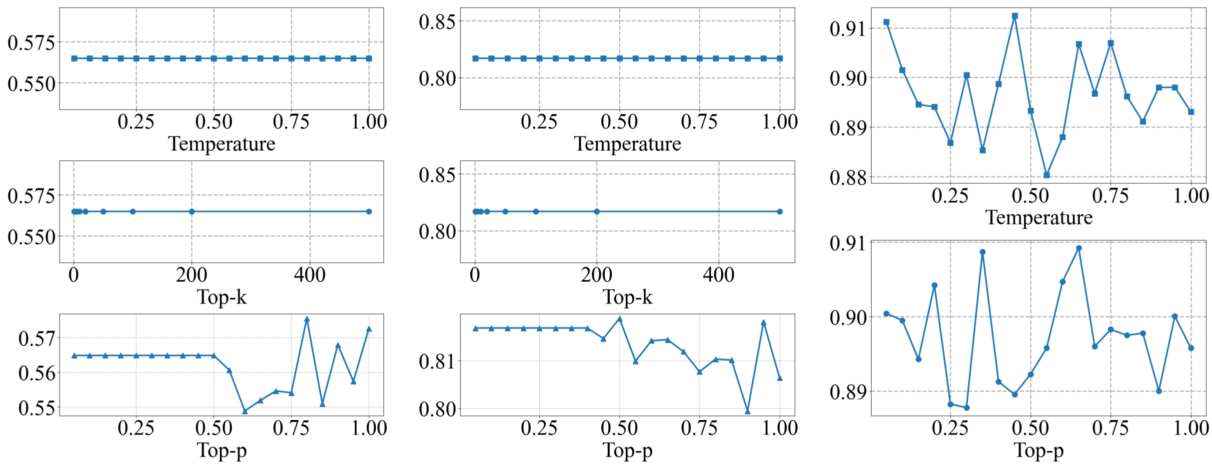
This result highlights **the position of each constraint within the prompt can substantially influence the model's output**. There's the need for careful consideration of keyword placement when designing prompt for LLMs. For example, placing critical constraints in positions that are more likely to be covered can significantly enhance the effectiveness of the model in downstream tasks.

3.2 Inherent Complexity of Compound Word

In previous experiments on position bias, we randomly shuffled keywords to mitigate the impact of specific words on final performance. In this experiment, we isolate the position bias and investigate the effect of different keywords on the final performance.

Experiments Setting From our observations in previous experiments, compound words often pose challenges in lexical processing. A compound word is formed from two or more words that collectively function as a single entity, such as "jellyfish" (a combination of "jelly" and "fish") and "anymore" (a combination of "any" and "more"). To evaluate the inherent complexity of compound words, we mixed 200 compound words with 200 random words, and conducted 5-keywords setting (i.e. generate a sentence with given five keywords) using LLaMA-13b-chat and GPT-4.

Experiment Result Our results show that LLaMA-13b-chat incorrectly split 65% of compound words and GPT4 split 42%, resulting in lower keyword coverage rates for compound words—35% for LLaMA-13b-chat and 58% for GPT4. In contrast, coverage for non-compound



(a) LLaMA2-7b-chat (b) LLaMA2-13b-chat (c) GPT-4

Figure 5: Comparison of decoding parameters across different models

words was significantly higher, at 74% for LLaMA-13b-chat and 92% for GPT4.We can conclude that **compound words have high inherent complexity in LCG tasks**, and it's more difficult to be covered by LLMs than non-compound words. This issue could be attributed to the subword tokenization methods used by these models, which may not effectively recognize and preserve the integrity of compound words.

The separation of compound words could not only result in unsatisfied constraints, but also lead to misinterpretations or significant alterations in the intended meaning of the output. For instance, when given the task of generating a sentence using the keywords: courthouse, build, and attract, the expected outcome is a sentence related to the criminal justice system. However, LLM split 'courthouse' into 'court' and 'house'. This leads to unintended interpretations, such as generating a sentence like, "*The basketball player hosted a tournament at the court built beside his house, attracting local talent to showcase their skills.*" Such a sentence completely deviates from the intended context of criminal justice.

3.3 Decoding Parameters

We notice that LLMs are usually evaluated for LCG tasks using only default decoding parameters(Zhang et al., 2023a), or limited fixed decoding parameters (Sun et al., 2023; Ashok and Poczos, 2024). We systematically varied decoding parameters to investigate the sensitivity of decoding parameters on lexical constraint generation. We aim to determine the impact of different decoding parameter settings on the performance of LLMs in LCG.

Experiment Setting Follow the prior practice (Huang et al., 2023), we experiment with the following three variants for decoding strategy:

- Temperature τ controls the sharpness of the next-token distribution. We vary it from 0.05 to 1 with step size 0.05.
- Top-K sampling filters the K most likely next words, and then the next predicted word will be sampled among these K words only. We vary K in $\{1, 2, 5, 10, 20, 50, 100, 200, 500\}$.
- Top-p sampling (Holtzman et al., 2019) chooses from the smallest possible set of words whose cumulative probability exceeds the probability p. We vary p from 0.05 to 1 with step size 0.05.

We evaluated all models under different decoding parameters in 10-keywords LCG task (i.e. generate sentence with given 10 keywords). Specifically, we only vary temperature and top-p parameters for GPT-3.5 and GPT-4, as we did not have control over the top-k settings.

Experiment Results Figure 5 presents the average keyword coverage rate for 150 instances, each containing 10 keywords (see Appendix B for more detail). For LLaMA2-7b-chat and LLaMA2-13b-chat, there appears to be no significant effect from variations in temperature and top-k settings, and the differences observed with various top-p settings are within a narrow 4% range, suggesting a low sensitivity to the top-p parameter. While GPT-4

Model	Recipe Generation			Table to Text			Profile Writing	
	n = 5	*n = 10*	*n = 15*	*n = 5*	*n = 10*	*n = 15*	*n = 5*	*n = 10*
LLaMA2-7b-chat	90%	21%	5%	87%	21%	21%	69%	28%
LLaMA2-13b-chat	89%	27%	17%	84%	45%	39%	73%	42%
GPT-3.5	90%	42%	54%	<u>97%</u>	80%	77%	90%	72%
GPT-4	**100%**	80%	45%	**100%**	<u>87%</u>	91%	<u>97%</u>	96%
LLaMA2-7b-chat (DnC-5)	<u>98%</u>	<u>99%</u>	<u>98%</u>	**100%**	**100%**	<u>99%</u>	**100%**	<u>99%</u>
LLaMA2-13b-chat (DnC-5)	**100%**	96%	94%	**100%**	**100%**	**100%**	**100%**	97%
GPT-3.5 (DnC-5)	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

Table 1: Results for LLMs' performance in real-word LCG task. The best results are highlighted in **boldface**, and the second-best results are <u>underlined</u>

demonstrates more variability under different settings, the difference between the highest and lowest scores remains confined to 4%.

This minimal variance suggests that the **decoding parameters are not highly sensitive for LLMs in LCG task, especially for temperature and top-k.**

4 Real-world applications

We have also evaluated the performance of LLMs in real-world applications to understand their practical effectiveness. In this section, we demonstrate three use cases: Recipe generation, table-to-text, and profile writing. We use the best decoding parameter configuration (*Top-p* = 0.9) identified in previous section for all following experiments. Example prompt and response for each application are attached to Appendix A.

4.1 Recipe Generation

The task is to generate a complete recipe given ingredients. LLMs need to create a coherent and structured set of cooking instructions that makes practical and culinary sense, and cover all provided keywords.

Experiment Setting. We randomly selected 100 food ingredients from the USDA National Nutrient Database (US Department of Agriculture, Agricultural Research Service, 2016) and grouped them into sets with varying numbers of ingredients (n = [5, 10,15]). Each group comprises ingredients versatile enough to be applicable to multiple recipes, guaranteeing the existence of at least one valid recipe for the given combination of ingredients. LLMs is then prompted in 3-shot fashion to generate recipe with given set of ingredients, where ingredients are keywords that are expected to be contained in the generated recipe. Each generated recipe is evaluate based on the instance success rate.

Evaluation Result. Table 1 presents the results of the experiment. When tasked with recipe generation, we observed that LLMs typically outline their plan in the initial sentence, such as "*Lemon Garlic Pasta is quick to prepare, making it perfect for a weeknight dinner yet elegant enough for entertaining guests.*", and "*To create Chicken and Mushroom Risotto, follow these steps*". **These introductory statements act as a double-edged sword.**

On the positive side, these introductory statements establish the scope for subsequent content generation, facilitating the model's ability to incorporate relevant keywords effectively. In the 5-keyword setting, the instance success rate for the LLaMA2 models increases by approximately 30% compared to Experiment 2.3, where LLMs were tasked solely with text generation under keyword constraints.

On the negative side, these introductory statements can detract from the final generation outcome if they are not accurate. If there are a large number of keywords, LLMs tend to include only a few in the first sentence, leading to the omission of the remaining keywords. As the number of keywords increases, there is a noticeable decline in performance across all models. For example, the instance success rate for LLaMA2-13b decreases from 89% to 17% as the number of constraints increases from 5 to 100.

4.2 Table to Text

Following previous work (Chen et al., 2023), table-to-text task takes a table as input, and formulate a table as a sequence of records. We evaluate the effectiveness of LLMs in presenting the essential information from the structured data in a narrative form.

Experiment Setting. WIKIBIO (Lebret et al., 2016) is a dataset contain of 728,321 tables data from English Wikipedia. We processed the WIKIBIO dataset by extracting keywords from each table's column headers as ground truth, and categorizing the tables into groups based on the number of keywords identified. For each group, 150 samples are randomly selected. Next, we construct instances from each group based on number of keywords needed. LLMs is then prompted in 3-shot fashion to summarize the content of these tables in a short paragraph, and each generated summary is evaluated based on the instance success rate.
Evaluation Result. As shown in table 1,GPT-4 demonstrates the strongest performance, achieving 100% accuracy with 5 keywords setting, and maintaining high instance success rate with larger number of keywords (87% for n = 10 and 91% for n = 15). However, other models, such as LLaMA2-7b-chat and LLaMA2-13b-chat, show notable declines in accuracy as the sample size increases, with significant drops from 87% to 21% and from 84% to 39%. This result indicates that LLMs struggle in satisfying more nuanced and complex constraints.

4.3 Profile Writing

Profile writing provides a quick overview of the client's basic information, significantly impacting decision-making and enhancing operational effectiveness. For instance, in healthcare, profiles summarize patient histories to guide treatment plans; in finance, they help assess client risk and customize financial services; and in the legal field, detailed client profiles are crucial for informed case management. This process can be viewed as a lexical constraint generation task, where the client's information acts as the constraint, and the resulting profile paragraph serves as the output.
Experiment Setting. This task is aimed to generate a profile contain all specific features of a client. We obtained data consists of various attributes of clients to assessing risk score, such as age, employment details, education, housing level, etc. In our experiment, we extract individual client information from this dataset, and prompt LLMs to generate a detailed profile graph contain all information.
Evaluation Result. Table 1 presents the results of the experiment. Similar to previous experiments, GPT-4 demonstrates the highest consistency and robustness among the models, scoring 97% with n = 5 and 96% with n = 10, showing only a slight decrease in performance with an increase in number of constraints. Other models show more significant drops in performance, denoting the need of improvement strategy.

Algorithm 1 Divide and Conquer Generation

$X \leftarrow$ set of all keywords
$K \leftarrow$ max number of iterations
$output \leftarrow$ empty sequence
$count \leftarrow 0$
while X is not empty **do**
  $sentence \leftarrow$ generate sentence with X
  $Y \leftarrow$ words in $sentence$
  $output \leftarrow$merge $(output, sentence)$
  $X \leftarrow X \setminus Y$
  if $count \geq K$ **then**
    return $output$
  end if
end while
return $output$

5 Divide and Conquer Generation

As demonstrated in previous experiments, LLMs face significant challenges in satisfying increasingly complex constraints. To address these difficulties, we propose a simple and effective strategy—Divide and Conquer Generation (DnC) —to improve LLMs' performance in Language Constraint Generation (LCG), which suitble for both white-box and black-box models.

5.1 Method

From our observation, we found LLMs struggle with complex tasks that encompass a large amount of keywords. In contrast, they exhibit a high success rate when dealing with simpler tasks involving a smaller number of keywords, which motivate us to break down the complex task to several simple tasks in divide and conquer fashion.

Algorithm 1 illustrates DnC strategy. Recall that the task is to generate a natural sentence containing the token sequence $Y = [y_1, y_2, \ldots, y_N]$ using a specified set of N keywords $X = [x_1, x_2, \ldots, x_N]$, such that $X \subseteq Y$. Our strategy iteratively generates sentences while addressing the missing keywords $X \setminus Y = \{x \in X \mid x \notin Y\}$ from each generation iteration, then merge these sentences into a cohesive final output. Figure 1 contains detailed example of the process of our strategy. We repeat this process until all constraints are satisfied, or exceed the max allowed number of iteration K.

5.2 Evaluation

Rejection Sampling (RJ) is a Monte Carlo algorithm to sample data from a sophisticated distribution with the help of a proxy distribution (Robert and Casella, 2004). This method can assist with black-box models, where texts that do not meet certain criteria are discarded, and the sampling process is iteratively repeated. We choose rejection sampling as the baseline method, and evaluate the DnC strategy.

We repeat the 15-keyword generation experiment with LLaMA2-7b-chat and GPT-3.5, using both RJ and DnC strategy under varying maximum number of iterations K allowed. Figure 6 demonstrate the result, where y-axis is the error rate in satisfying all lexical constraints (i.e. 1 minus the instance success rate). At $K = 0$, the models generate in a vanilla setting, without employing any specific strategies. From the result, we can observe that while the RJ strategy manages to reduce the error rate, it does not lead to significant improvements. In the contrast, DnC help both model achieve a near-perfect performance (error rate close to 0%) with $K = 4$. With the help of DnC, LLaMA2-7b-chat model decrease error rate from approximately 96% to 3%, demonstrating the effectiveness of the DnC approach.

Furthermore, we revisited application tasks introduced in 4. Table 1 compares the instance success rates for each approach. From the result, with the implementation of the DnC strategy, all models achieve near-perfect performance (instance coverage rates approaching 100%). Specifically, the LLaMA2-7b-chat model records an average improvement of 61% across all tasks with the help of DnC strategy. Notably, GPT-3.5 (DNC-5) achieves a 100% instance success rate for all tasks.

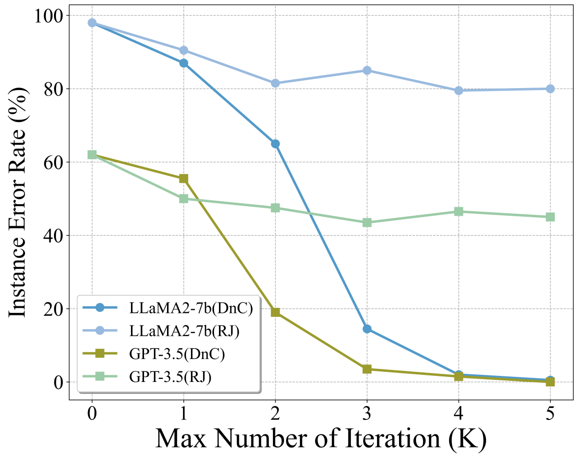


Figure 6: Comparison experiment of Rejection sampling (RJ) and Divide-and-Conquer Generation (DnC). x-axis is the max number of iteration allowed, and y-axis is the error rate of each approach in satisfying all lexical constraints.

6 Related Work

LLMs Evaluation With recent advancements in Large Language Models (LLMs), there is increasing interest in evaluating controllable text generation tasks. Sun et al. conducted evaluations of these tasks and discovered that LLMs often struggle to meet fine-grained constraints. However, their analysis of lexical constraint generation was limited to relatively simple constraints in a narrow context. Our work expands on this by conducting a more comprehensive and in-depth analysis of lexical constraint generation, providing deeper insights into the capabilities and limitations of LLMs in this area. Additionally, they have not propose solution but we did.

Lexical Constrained Generation There are many works trying to improve lexical constrained generation. We roughly categorize these studies: (1) proposing decoding strategy: Grid Beam Search tweaked the beam search algorithm to meet lexical constraints by increasing the weights for the constraint lexicons during the beam search (Hokamp and Liu, 2017). (2) specialized model structure: InsNET is an expressive insertion-based text generator with efficient training and flexible decoding (Lu et al., 2022). However, they are not suitable for recent pre-trained LLMs due to the black-box nature. There are only a few studies focus on the prompt-based approach (Iso, 2022), and they failed to show effectiveness in real-world applications with modern LLMs.

7 Conclusion

We conduct in-depth systematically analysis on LLMs in satisfying lexical constraints, and identify the current challenges faced by LLMs in satisfying lexical constraints, including (1) position bias, where LLMs tend to satisfy constraints that appear in specific positions within the input; (2) the inherent complexity of compound words; and (3) insensitive decoding parameters, which minimally impact the performance of LLMs in LCG. Additionally, we provide a effective solution to these challenges, namely DnC generation, paving the way for more sophisticated downstream applications.

Limitation

Our work is not without limitations. First, our experiments are prompt-based, requiring extensive prompt engineering effort. While we selected the best-performing prompts available, there remains the possibility that more effective prompts could further enhance the reported result. Second, automatic evaluations have inherent imperfections. Third, the proposed Divide and Conquer (DnC) strategy increases the number of API calls, resulting in higher costs. Additionally, due to the nature of the DnC approach, the coherence of the merged results may be compromised. We leave the evaluation and further refinement of this to future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Dhananjay Ashok and Barnabas Poczos. 2024. Controllable text generation in the instruction-tuning era. arXiv preprint arXiv:2405.01490.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Miao Chen, Xinjiang Lu, Tong Xu, Yanyan Li, Jingbo Zhou, Dejing Dou, and Hui Xiong. 2023. Towards table-to-text generation with pretrained language model: A table structure understanding and text deliberating approach. arXiv preprint arXiv:2301.02071.

Helena H. Lee, Ke Shu, Palakorn Achananuparp, Philips Kokoh Prasetyo, Yue Liu, Ee-Peng Lim, and Lav R Varshney. 2020. Recipegpt: Generative pretraining based cooking recipe generation and evaluation system. In Companion Proceedings of the Web Conference 2020, pages 181–184.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source llms via exploiting generation. arXiv preprint arXiv:2310.06987.

Hayate Iso. 2022. Autotemplate: A simple recipe for lexically constrained text generation. arXiv preprint arXiv:2211.08387.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In Conferences of the Association for Machine Translation in the Americas: MT Researchers' Track, pages 107–120, Austin, TX, USA. The Association for Machine Translation in the Americas.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. arXiv preprint arXiv:1603.07771.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2019. Commongen: A constrained text generation challenge for generative commonsense reasoning. arXiv preprint arXiv:1911.03705.

Sidi Lu, Tao Meng, and Nanyun Peng. 2022. Insnet: An efficient, flexible, and performant insertion-based text generation model. Advances in Neural Information Processing Systems, 35:7011–7023.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, et al. 2021. Neurologic a* esque decoding: Constrained text generation with lookahead heuristics. arXiv preprint arXiv:2112.08726.

Tao Meng, Sidi Lu, Nanyun Peng, and Kai-Wei Chang. 2022. Controllable text generation with neurally-decomposed oracle. Advances in Neural Information Processing Systems, 35:28125–28139.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35:27730–27744.

Jing Qian, Li Dong, Yelong Shen, Furu Wei, and Weizhu Chen. 2022. Controllable natural language generation with contrastive prefixes. arXiv preprint arXiv:2202.13257.

Christian P. Robert and George Casella. 2004. Monte Carlo Statistical Methods. Springer Texts in Statistics. Springer.

Lei Sha. 2020. Gradient-guided unsupervised lexically constrained text generation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 8692–8703, Online. Association for Computational Linguistics.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 3155–3168, Singapore. Association for Computational Linguistics.

US Department of Agriculture, Agricultural Research Service. 2016. Usda national nutrient database for standard reference, release 28 (slightly revised). http://www.ars.usda.gov/nea/bhnrc/mafcl. Version Current: May 2016.

Yiwei Wang and etc. 2023. Primacy effect of chatgpt. arXiv preprint arXiv:0000.00000.

Kexin Yang, Dayiheng Liu, Wenqiang Lei, Baosong Yang, Mingfeng Xue, Boxing Chen, and Jun Xie. 2022. Tailor: A prompt-based approach to attribute-based controlled text generation. arXiv preprint arXiv:2204.13362.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023a. A survey of controllable text generation using transformer-based pre-trained language models. ACM Computing Surveys, 56(3):1–37.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023b. Instruction tuning for large language models: A survey. arXiv preprint arXiv:2308.10792.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. In International Conference on Machine Learning, pages 42602–42613. PMLR.

A Appendix: Prompts

Prompt and example response for lexical constrained generation, and downstream applications.

A.1 Example of Lexical Constrained generation

Prompt Generate a sentence using the following keywords: cat, leaves, sunny, leaps, energy

Response Generate a sentence using the following keywords: cat, leaves, sunny, leaps, energy

A.2 Example of Recipe Generation

Prompt Generate a recipe using the following ingredients: milk, beef, rice.

Response The recipe for Creamy Beef and Rice Casserole: Begin by cooking 1 cup of rice according to package instructions and set aside to cool. Add 1 pound of ground beef to the skillet, breaking it apart with a spatula. Cook until the beef is thoroughly browned and no pink remains. Drain any excess fat and set aside.In a large casserole dish, combine the cooked rice, browned beef, and 2 cups of milk, mixing well. Stir in 1 cup of shredded cheddar cheese, season with salt and freshly ground black pepper to taste, and add a dash of paprika for a smoky flavor, if desired. Dot the top of the mixture with small pieces of butter to enrich the flavor.Preheat your oven to 350°F (175°C). Cover the casserole dish with aluminum foil and bake for 25 minutes. Remove the foil and continue baking for an additional 5-10 minutes, or until the top is bubbly and golden brown. Remove from the oven and let it sit for a few minutes to settle

A.3 Example of Table-to-text

Prompt Generate a short paragraph to summarize the table content:

Nationality	German
Birth Date	1954
Name	Walter Extra
Article Title	Aviation Legend: Walter Extra
Occupation	Aircraft Designer and Manufacturer

Response Walter extra is a german award-winning aerobatic pilot, chief aircraft designer and founder of extra flugzeugbau (extra aircraft construction)), a manufacturer of aerobatic aircraft . extra was trained as a mechanical engineer. Walter extra has designed a series of performance aircraft which include unlimited aerobatic aircraft and turboprop transports.

A.4 Example of Profile Writting

Prompt Write a profiling paragraph contains all features of the client: Name: Ben Smith; Age: 29; Gender: Male; Race: White; Marriage: Single; FICO:740; Education: Bachelor; Occupation: Engineer; Location: Chicago; Housing: Rent.

Response Ben Smith is a 29-year-old white male, currently single, residing in Chicago. He holds a Bachelor's degree and works as an engineer, a profession that demands high technical proficiency and strong problem-solving skills. Ben rents his housing, which could suggest flexibility in his living arrangements or a preference for less permanent commitments in terms of residency. With a FICO score of 740, Ben demonstrates strong financial responsibility, indicating good management of credit and other financial obligations. This profile suggests that Ben is both stable in his career and adaptable in his personal life.

B Appendix: Decoding Hyperparameters

We visualize the instance success rate for each instance under different decoding hyperparameters. Heatmaps are attached as following.

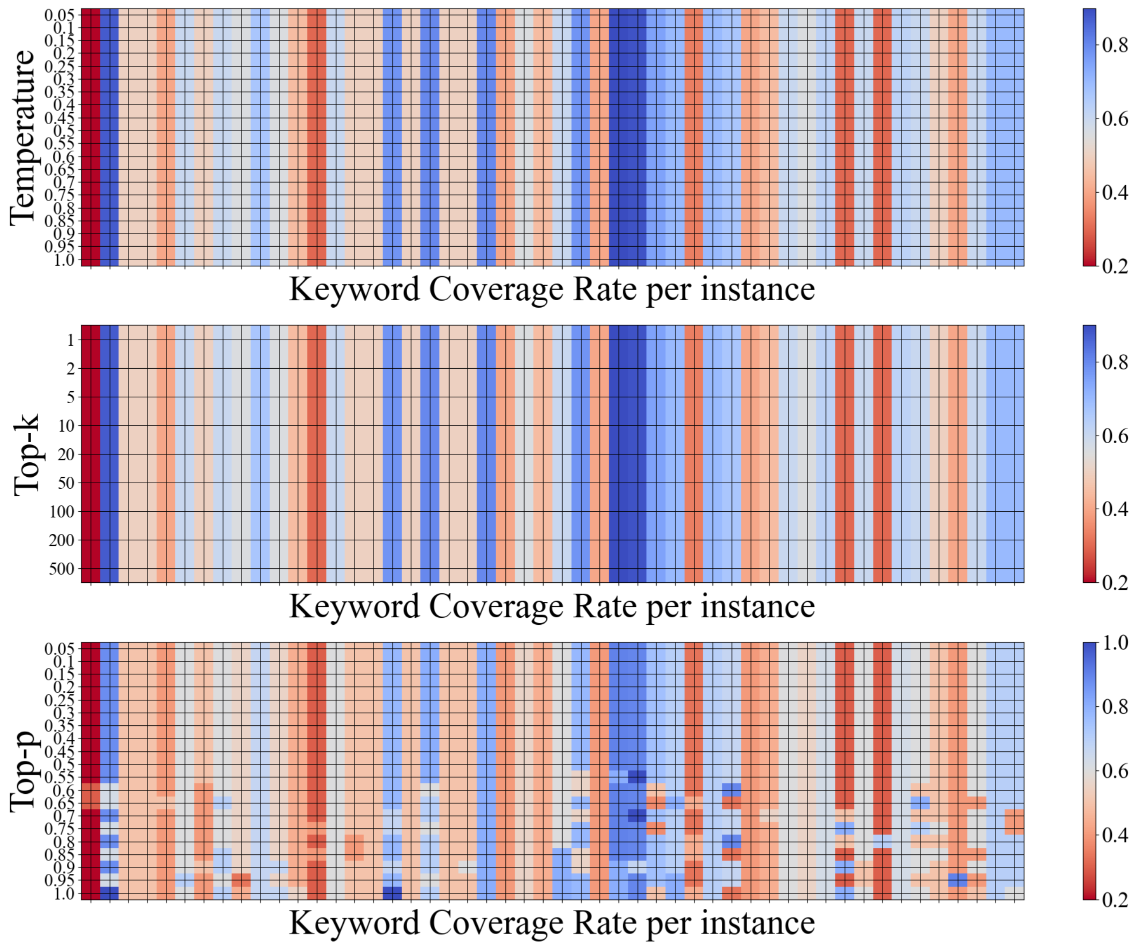


Figure 7: Heatmap of keyword coverage rate per instance for LLaMA2-7b model.

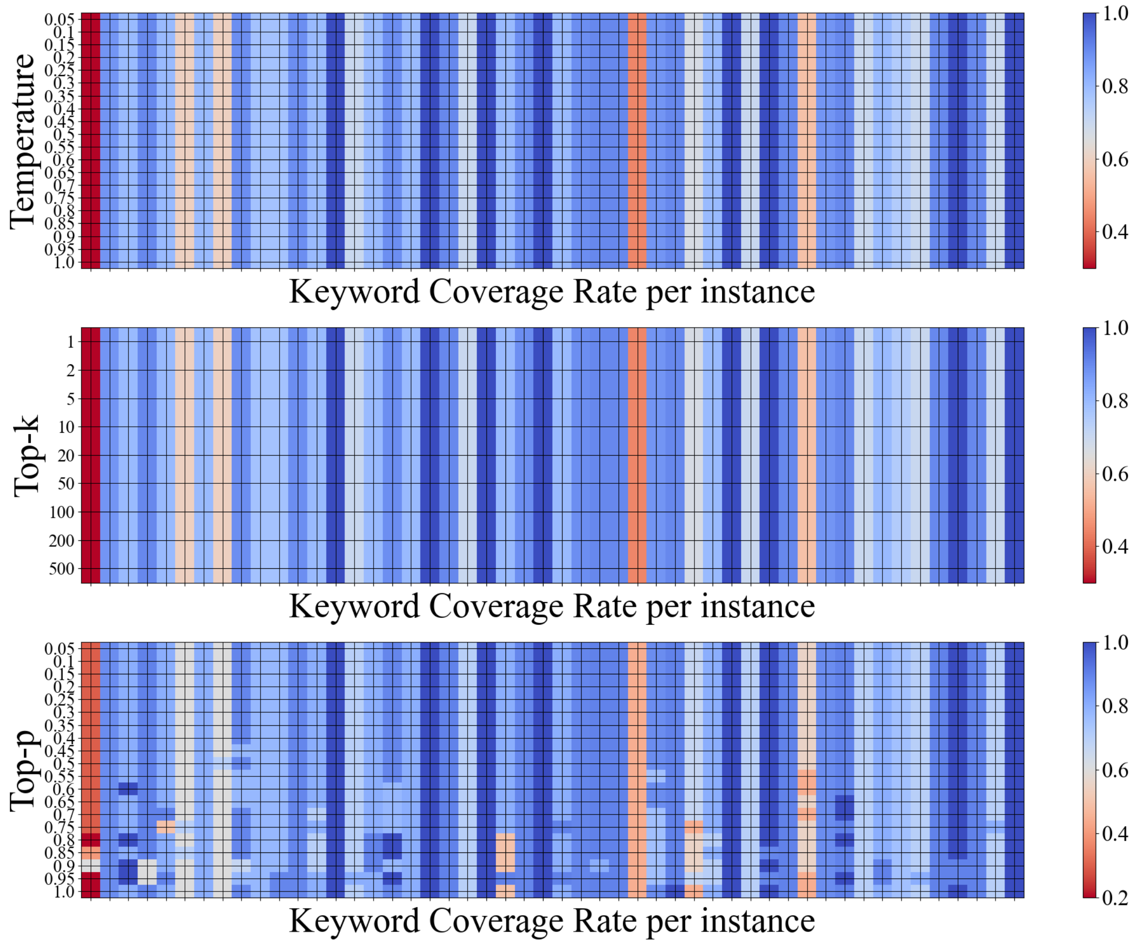


Figure 8: Heatmap of keyword coverage rate per instance for LLaMA2-13b model.

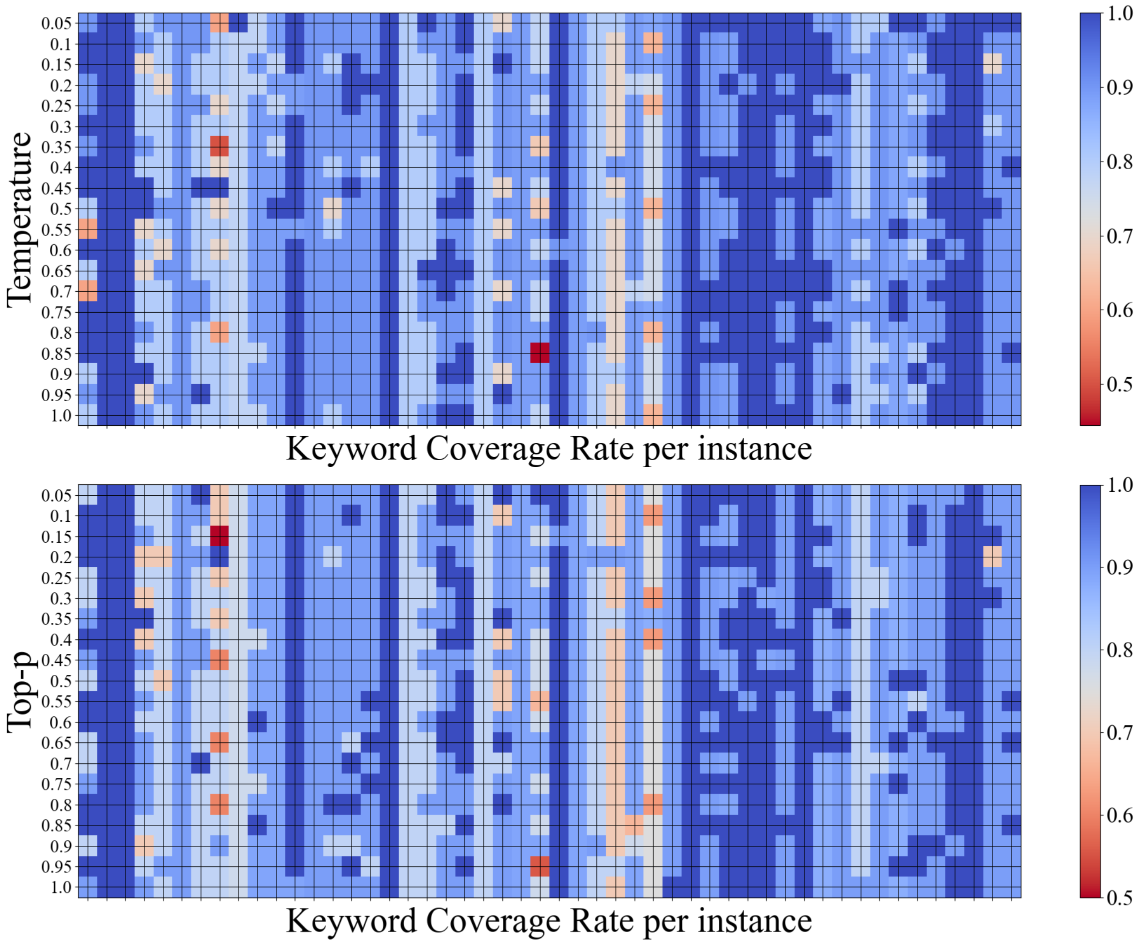


Figure 9: Heatmap of keyword coverage rate per instance for GPT model.

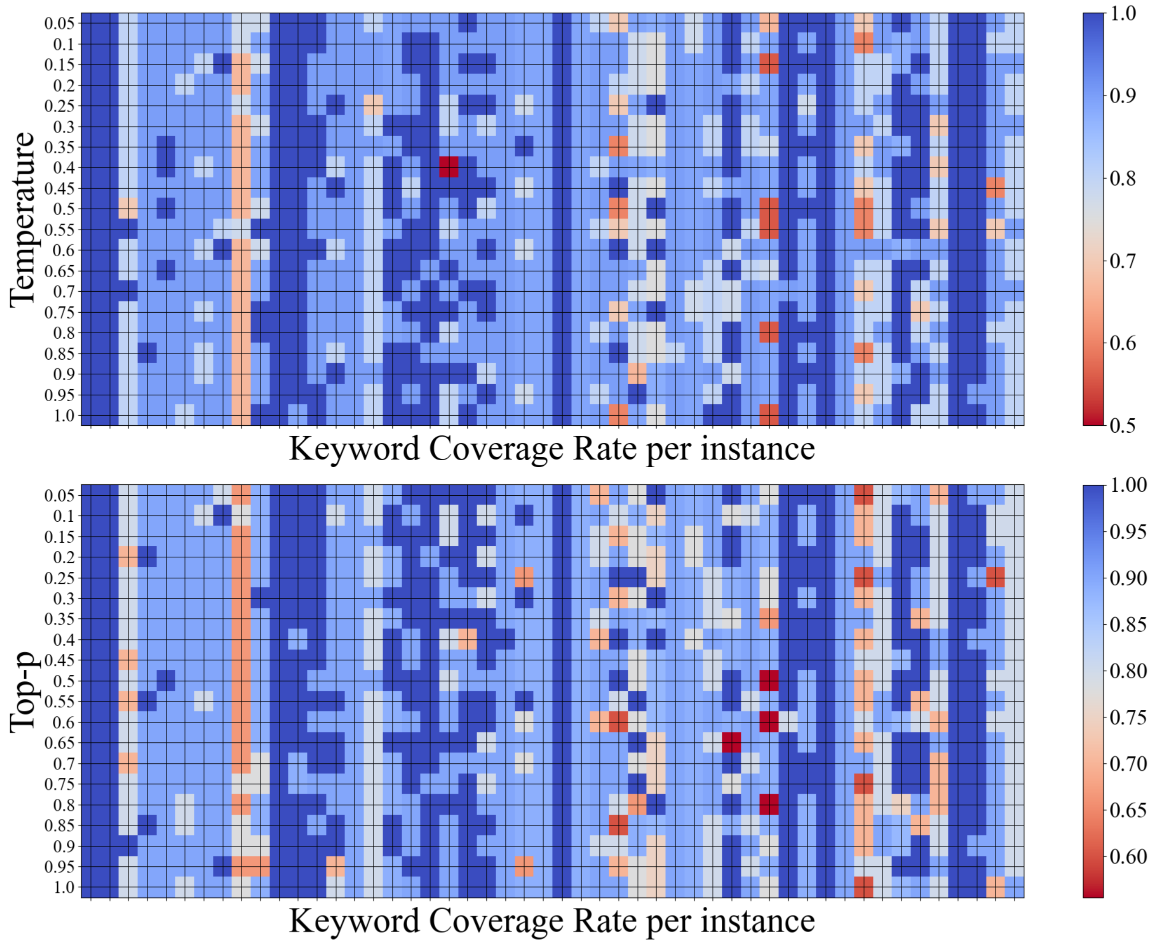


Figure 10: Heatmap of keyword coverage rate per instance for GPT-4 model.