# Zero-Shot Keyphrase Generation: Investigating Specialized Instructions and Multi-Sample Aggregation on Large Language Models

**Jayanth Mohan*** **Jishnu Ray Chowdhury*†** **Tomas Malik** **Cornelia Caragea**
Computer Science
University of Illinois Chicago
jmoha11@uic.edu jraych2@uic.edu tmalik6@uic.edu cornelia@uic.edu

## Abstract

Keyphrases are the essential topical phrases that summarize a document. Keyphrase generation is a long-standing NLP task for automatically generating keyphrases for a given document. While the task has been comprehensively explored in the past via various models, only a few works perform some preliminary analysis of Large Language Models (LLMs) for the task. Given the impact of LLMs in the field of NLP, it is important to conduct a more thorough examination of their potential for keyphrase generation. In this paper, we attempt to meet this demand with our research agenda. Specifically, we focus on the zero-shot capabilities of open-source instruction-tuned LLMs (Phi-3, Llama-3) and the closed-source GPT-4o for this task. We systematically investigate the effect of providing task-relevant specialized instructions in the prompt. Moreover, we design task-specific counterparts to self-consistency-style strategies for LLMs and show significant benefits from our proposals over the baselines.

## 1 Introduction

Keyphrases are concise, representative phrases that encapsulate the most essential and relevant topical information in a document (Hasan and Ng, 2014). They serve as a high-level summary, providing quick insight into the text. Keyphrases can be "present" if they appear verbatim in the text, or "absent" if they are semantically implied and do not occur explicitly in the text. While keyphrase extraction focuses on identifying present keyphrases (Park and Caragea, 2023; Patel and Caragea, 2021; Al-Zaidy et al., 2019; Bennani-Smires et al., 2018; Yu and Ng, 2018; Florescu and Caragea, 2017; Sterckx et al., 2016; Gollapalli and Caragea, 2014), keyphrase generation (KPG) extends the task to include both present and absent keyphrases (Garg et al., 2023; Chowdhury et al., 2022; Garg et al., 2022; Meng et al., 2017; Yuan et al., 2020; Chan et al., 2019; Chen et al., 2020). Recent advancements in keyphrase research, including this work, focus primarily on KPG, as it provides a more comprehensive summary of the document's information. Keyphrases are vital in various information retrieval and NLP applications, such as document indexing and retrieval (Jones and Staveley, 1999; Boudin et al., 2020), summarization (Wang and Cardie, 2013; Abu-Jbara and Radev, 2011), content recommendation (Augenstein et al., 2017), and search engine optimization (Song et al., 2006).

Various previous approaches have attempted to tackle KPG. Most of them are sequence-to-sequence approaches that are trained from scratch specifically for KPG (Meng et al., 2017; Yuan et al., 2020; Chan et al., 2019; Chen et al., 2020; Ye et al., 2021b; Thomas and Vajjala, 2024). More recently, some approaches explore finetuning of pre-trained language models such as BART or T5 for KPG (Wu et al., 2021; Kulkarni et al., 2022; Wu et al., 2023, 2024a; Choi et al., 2023). However, the field of Natural Language Processing (NLP), on the other hand, is moving away from such approaches and towards the utilization of Large Language Models (LLMs) (Iyer et al., 2022; Touvron et al., 2023) that typically have much higher parameters and are pre-trained on larger scale datasets. As such, naturally, there is a question as to how well such models can be operated towards KPG. A few prior works conduct some studies to answer this question, primarily investigating ChatGPT as a zero-shot generator. However, they are only preliminary studies that investigate a few variants of prompts (Song et al., 2023b,a; Martínez-Cruz et al., 2023). Our work aims to extend such studies further. Specifically, in this paper, we aim to answer three research questions (RQ1, RQ2, RQ3) as defined below.

**RQ1:** *Can LLMs be guided to focus specifically on*

*Both authors contributed equally to this research.

†Most work done at the University of Illinois Chicago unaffiliated to the author's current position at Bloomberg.



*Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7882–7899
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

*present or absent keyphrases via prompts?*

As discussed before, KPG typically involves the generation of two distinct types of keyphrases—present and absent which may require distinct strategies. In Song et al. (2023b), we also find that the same prompt is not necessarily good at both present and absent generation simultaneously. Thus, the question arises if we can create separate "specialist" prompts - one specializing in present keyphrase generation and another specializing in absent keyphrase generation. If this succeeds, we can come up with a way to combine the specialists' results to improve both present and absent keyphrase generation performance. We describe our designed specialist prompts in §2.2 and show their corresponding evaluation in §3.2.

**RQ2:** *Do more specific instructions about controlling the number of keyphrases and/or the order of generation help LLMs?*

In our baselines, we provide basic instructions regarding formatting to enable parsing of keyphrases through downstream post-processing methods. However, there is a potential to explore the application of more detailed instructions to the models. For example, we might want to specify how we want the keyphrases to be ordered - such as most relevant keyphrases being generated before less relevant ones. Metrics such as ($F_1$@5) used in keyphrase generation, focus on some first $k$ keyphrases, so it is important for the LLMs to generate the best keyphrases first. We might also want to instruct the model more specifically to not overgenerate. We find that LLMs tend to generate more keyphrases on average compared to other smaller models, which can lower precision. We design specific instructions corresponding to these points in §2.3 and experimentally investigate them in §3.3.

**RQ3:** *Can multiple samplings of an LLM from the same input prompt be leveraged to improve performance in keyphrase generation?*

Often in KPG, beam search is used to create multiple sequences of keyphrases to improve the recall of keyphrases (Chowdhury et al., 2022; Thomas and Vajjala, 2024; Yuan et al., 2020). On the other hand, the use of multiple samplings has been successful with LLMs in general NLP tasks as well. For instance, the self-consistency strategy leverages majority voting (or other aggregation techniques) across multiple sampled results for a question to boost the performance of LLMs (Wang et al., 2023). Given the success of self-consistency (on general NLP tasks) and beam search (for KPG), we raise the question if we can similarly leverage multiple sampling from LLMs for KPG specifically. To answer this question, we devise various multi-sampling aggregation strategies for KPG in §2.4 and demonstrate their corresponding results experimentally in §3.4.

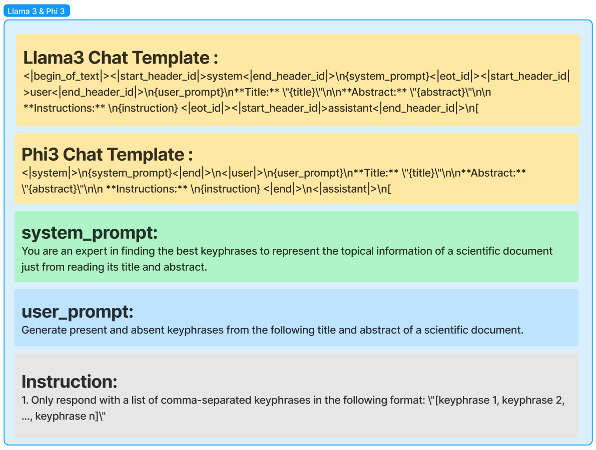


Figure 1: Baseline template used for Llama-3 and Phi-3.

We focus primarily on open-source, instruction-tuned models, specifically LLama-3 and Phi-3, in a zero-shot setting. Additionally, we include experiments with GPT-4o to benchmark against a bigger closed-source model. In our experiments, we find that specialist prompts for our models do not help (answering RQ1 negatively) and that additional detailed instructions do not help consistently (answering RQ2 negatively). However, we find that multi-sampling can be successfully leveraged to substantially boost the performance of LLMs for KPG (answering RQ3 affirmatively).

## 2 Method

We explore the performance of two open-source instruction-tuned LLMs - **Llama-3.0 8B Instruct** (Dubey et al., 2024) and **Phi-3.0 3.8B Mini 128K Instruct** (Abdin et al., 2024) and one closed-source LLM - **GPT-4o** (version: gpt-4o-2024-11-20) (Achiam et al., 2023) on KPG for five different datasets in a zero-shot setting. We explain our main approaches in the following subsections.

### 2.1 Baseline

Here, we explain the construction of baseline prompts for Llama-3, Phi-3 and GPT-4o. First, we keep their prompt templates consistent with their corresponding chat templates as shown in

Figure 1. Note that at the end of the prompt, we leave an open parenthesis “[” so that the models can directly start generating the keyphrases without any in-between irrelevant text. As can be seen in the chat templates, there are five variables: 1) the `system_prompt`, 2) the `user_prompt`, 3) the `instruction`, 4) the `title`, and 5) the `abstract`. The last two are inputs from the dataset, whereas the first three are manually defined. We define them the same way for Llama-3 and Phi-3. For GPT-4o, we use the chat completion API for sending the system prompt and user prompt. We skipped the open parenthesis for the assistant role in GPT-4o because the provided chat completion API does not support partial conversational turns. Our definitions for `system_prompt`, `user_prompt`, and `instruction` variables are also shown in Figure 1. When evaluating the models on KP-Times, we changed any occurrence of “scientific document” with “news article”. The user prompt is roughly inspired from the TP4 prompt template in Song et al. (2023b)[1]. We use TP4 because it presents a reasonable balance in their paper. In the baseline, the `instruction` merely provides some formatting specifications to make parsing of the keyphrases lists easier.

### 2.2 Specialist Prompts (RQ1)

As discussed before, the same prompt may not be the best for both present and absent keyphrase generation. As such, we consider if we can improve present performance and absent performance separately with “specialist” prompts - one dedicated to present keyphrase extraction and another to absent keyphrase generation. We design the present specialist prompt by simply changing the baseline `user_prompt` to `“Extract present keyphrases from the following title and abstract of a scientific document.”` Similarly, we design the absent specialist prompt by simply changing the baseline `user_prompt` to `“Generate absent keyphrases from the following title and abstract of a scientific document.”` This results in the creation of two separate prompts which we test separately.

### 2.3 Additional Instructions (RQ2)

We consider here whether LLMs can benefit from more specific instructions as to how to order keyphrases and how many keyphrases to generate. We consider two types of instructions:

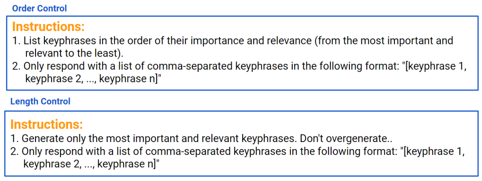


Figure 2: Instructions used for Order Control and Length Control. Note that the main values for the instruction variable are in the blue bordered box. The differences of box sizes and colours are for visualization only and do not play any role in the actual prompt.

**1. Order Control Instruction:** As we discussed before, the order of the keyphrases can be relevant, especially for metrics like $F_1@5$ where only the first 5 keyphrases are kept, and we are interested in keeping the best keyphrases within the first few. So we experiment with an additional instruction that explicitly specifies the model to order the keyphrases in the descending order of relevance and importance. Concretely, we do this by changing the value of the `instruction` variable from the baseline into a numbered list having both the formatting instruction and the order control instruction as shown in Figure 2.

**2. Length Control Instruction:** Here we focus on reducing overgeneration, which can negatively impact metrics like precision. For this, we instruct the model to generate only the most relevant keyphrases, avoiding unnecessary additions. Concretely, we do this, similar to above, by changing the `instruction` variable from the baseline as shown in Figure 2.

**3. Combined Control:** Here we integrate both the Order Control and Length Control instructions to prime the model to generate a concise list of keyphrases ordered by relevance. We do this by adding both the Order Control and Length Control instructions into the numbered list of instructions similar to before. In our implementation, the Length Control instruction is the first instruction (number 1), the Order Control instruction is the second one (number 2), and the formatting instruction from the baseline is the third one (number 3).

### 2.4 Multi-Sampling (RQ3)

To investigate RQ3, we stochastically generate multiple samples from LLMs with the baseline prompt using different temperatures for diversity. We take independent samples similar to self-consistency

[1] Similar to Song et al. (2023b), we also verified that LLama-3 and Phi-3 can distinguish the meaning of present and absent keyphrases by themselves. Thus, we did not present any further overt definition of them in the prompts.

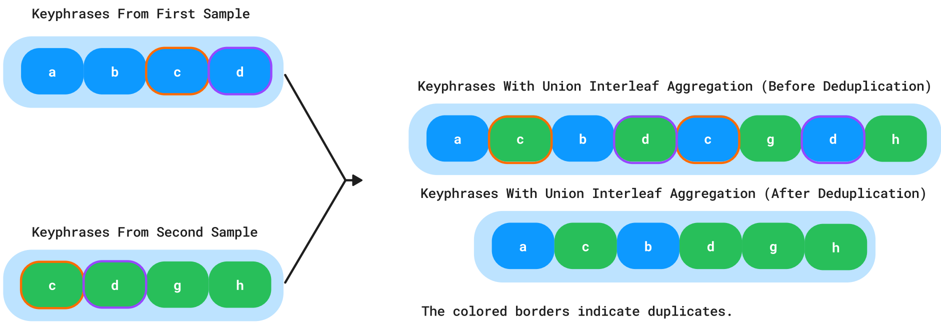


Figure 3: Visualization of Union Interleaf aggregation over multiple samples.

strategies (Wang et al., 2023). In addition, in Appendix B, we show that beam search, which is often used in KPG, does not improve performance on KPG, and at the same time can become expensive with large LLMs and tends to have worse diversity.

In our multi-sampling context, for a specific input, we initially end up having a list of samples as an answer: $S = (S_1, S_2, S_3, \ldots, S_n)$. Here $n$ is the number of samples. Each sample $S_i$ is a sequence of keyphrases: $S_i = (k_1^i, k_2^i, k_3^i, \ldots, k_m^i)$. Each keyphrase ($k_j^i$) is a string. We describe our pipeline for processing such samples below.

**Ranking Samples:** We first sort the generated samples before applying any aggregation strategy in the ascending order of their perplexity. We do this because some of our aggregation techniques (e.g., Union Concatenation that we discuss below) is biased towards putting the keyphrases of the earlier samples in $S$ earlier. As we discussed before, the order of the keyphrases can be relevant for metrics like $\text{F}_1$@5. Thus, we sort them to keep the "best" samples according to perplexity at the forefront for any downstream aggregation.

**Keyphrase Normalization:** Before aggregation, we also normalize the keyphrases using standard techniques - such as lower-casing and stemming. These are standard normalization strategies also used for evaluation to determine which keyphrases are identical. We also deduplicate each sample while preserving the order.

**Keyphrase Aggregation Strategies:** After ranking and normalization is done, the question is how to aggregate the results. We devise several strategies for aggregating the results from different samples that we discuss below.

1. *Union:* This is a simple strategy, where we treat all the generated lists of keyphrases ($S_i$) as sets and apply union operation. The result is $\cup_{i=1}^{n} S_i$. All order information is destroyed in this process.

2. *Union Concatenation:* In the context of KPG, a typical method used during beam-search to aggregate the results from multiple beams is to concatenate each of the beam sequences together (starting from the highest-ranked beam to the lowest). We simulate the same strategy here with Union Concatenation. In this approach, we concatenate all the samples: $||_{i=1}^{n} S_i$ (here $||$ denotes concatenation operator). After that, we deduplicate the concatenated sequence in an order-preserving manner (the first occurrence of a duplicate is the one that remains).

3. *Union Interleaf:* In this strategy, we initially combine the samples in an interleaving pattern. That is, first we take all the first keyphrases from each sample, then all the second keyphrases from each sample, and so on. We add them to a combined list in that order. The combined list will look like: $(k_1^1, k_1^2, \ldots, k_1^n, k_2^1, k_2^2, \ldots k_2^n, k_m^1 \ldots, k_m^n)$. After this, we perform an order-preserving deduplication as in Union Concatenation. The visualization of this process is provided in Figure 3.

4. *Frequency Order:* Frequency Order is the closest counterpart to majority voting as applicable for KPG. In this method, we consider the frequency of occurrence for each normalized keyphrase across all the samples. Then we sort the keyphrases in descending order of their frequency of occurrence. Thus, the highest "voted" (most frequent) keyphrase gets to be at the forefront of the aggregated list getting the maximum preference. In case of ties, we follow the order in Union Interleaf. That is, if there is a tie in terms of frequency between $k_1$ and $k_2$, then $k_1$ should come ahead of $k_2$ if and only if it occurs before $k_2$ in the union interleaf result for the same samples.

**Dynamic Keyphrase Number Selection:** Once the aggregation is done, there is a separate ques-

| | Inspec | | Krapivin | | SemEval | | KP20K | | KPTimes | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** |
| **Present Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | **48.3** | **40.5** | 30.9 | 32.4 | **35.5** | **36.2** | 27.7 | **30.7** | **27.0** | **31.3** |
| Present Specialist | 46.9 | 40.2 | 30.6 | 31.5 | 34.8 | 33.6 | **29.0** | 30.4 | 24.0 | 29.3 |
| Absent Specialist | 47.9 | **40.5** | **31.6** | **32.8** | 35.4 | 36.0 | 28.2 | **30.7** | 22.6 | 29.5 |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | 48.2 | 42.2 | 22.2 | 22.5 | **28.4** | **28.6** | 17.6 | **19.1** | **9.3** | **11.2** |
| Present Specialist | **48.4** | **42.6** | **22.6** | 22.6 | 26.3 | 26.0 | 17.6 | 19.0 | 8.7 | 10.5 |
| Absent Specialist | 46.6 | 41.2 | **23.5** | **22.9** | 27.8 | 28.5 | **18.2** | **19.1** | 8.1 | 9.0 |
| **GPT-4o** | | | | | | | | | | |
| Baseline | 56.8 | 49.7 | 26.0 | **28.0** | 33.2 | 34.1 | 20.1 | 24.7 | 11.4 | 14.7 |
| Present Specialist | **57.5** | **50.1** | **27.1** | **28.0** | **33.6** | **34.3** | **20.6** | **24.3** | **11.9** | **15.7** |
| Absent Specialist | 37.6 | 33.8 | 23.3 | 22.9 | 24.8 | 24.9 | 15.9 | 17.0 | 7.5 | 7.9 |
| **Absent Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | **6.8** | **5.5** | **4.6** | **3.8** | **3.2** | **3.0** | 3.8 | 3.0 | **4.6** | **3.6** |
| Present Specialist | 5.6 | 4.5 | 3.4 | 2.6 | 2.5 | 2.1 | 3.4 | 2.7 | 4.1 | 3.3 |
| Absent Specialist | 6.4 | 5.0 | 4.2 | **3.8** | 3.1 | **3.0** | **4.0** | **3.2** | 4.2 | **3.6** |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | **7.3** | **6.3** | **1.3** | 1.1 | **2.0** | **1.5** | 1.3 | 1.1 | **0.4** | **0.4** |
| Present Specialist | 7.0 | 5.6 | 1.2 | **1.2** | 1.6 | 1.3 | 1.3 | 1.1 | **0.4** | **0.4** |
| Absent Specialist | 6.6 | 5.5 | **1.3** | **1.2** | 1.7 | 1.4 | **1.4** | **1.2** | 0.3 | 0.4 |
| **GPT-4o** | | | | | | | | | | |
| Baseline | 10.6 | 10.6 | **4.0** | **4.0** | 2.6 | 2.6 | 2.4 | 2.5 | 0.4 | 0.5 |
| Present Specialist | **12.4** | **12.4** | 3.0 | 3.0 | 2.8 | 2.5 | 2.5 | 2.5 | **0.8** | **0.8** |
| Absent Specialist | 6.5 | 6.5 | 3.5 | 3.5 | **5.0** | **4.4** | **3.2** | **3.4** | 0.5 | 0.6 |

Table 1: Comparison of baseline prompts and specialist prompts for present and absent keyphrase generation.

tion as to how to dynamically select an appropriate number of keyphrases for each input. Normally, in the baseline single sample setting, we can simply use all the keyphrases predicted by the model until the end of sequence marker. However, with increasing number of samples being aggregated, the total keyphrases can become arbitrarily high. This can lead to the overgeneration of noisy keyphrases - leading to degraded precision and $F_1$, especially for @M metrics (which considers all keyphrases by the model not just some top $k$). To resolve this, we devise an automatic protocol to dynamically select a variable number of present keyphrases and a variable number of absent keyphrases from the total generation. Concretely, we first calculate the average number of present keyphrases (say $M_{pre}$) and average number of absent keyphrases (say $M_{abs}$) per sample for a specific input.[2] Then from the aggregated list, we take the first $M_{pre}$ present keyphrases and the first $M_{abs}$ absent keyphrases. We treat this as the final model prediction for $F_1$@M metric calculation.

**Discussion:** A problem with Union Concatenation is that it can lead to ignoring later samples altogether due to truncating the concatenation based on either top-5 selections (for $F_1$@5) or top $M_{pre}$ and $M_{abs}$ selections (for $F_1$@M). It can be still a reasonable strategy if the concatenation is ordered such that the first few samples are of higher quality, but even with our perplexity-based ranking, it is unlikely to have that much of a difference in quality among the samples given that they are each sampled independently based on the same process. Moreover, it can be the case, that earlier keyphrases from later samples are of higher quality than later keyphrases of earlier samples. This can happen if LLMs generate the most relevant keyphrases first. Union Concat would not respect this factor. Union Interleaf or Frequency Order based aggregations, on the other hand, can address some of these points much better in theory - resulting in a better intermingling of different samples in the final list.

## 3 Experiments and Results

For our experiments, we choose a temperature of $0.8$ which we use consistently[3] across all models

[2]In case the average is not a whole number, we take the ceiling.

[3]We chose $0.8$ because it is in the standard range of temperature typically used for self-consistency for diverse multi-

| Models | Inspec F1@M | Inspec F1@5 | Krapivin F1@M | Krapivin F1@5 | SemEval F1@M | SemEval F1@5 | KP20K F1@M | KP20K F1@5 | KPTimes F1@M | KPTimes F1@5 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Present Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | **48.3** | **40.5** | 30.9 | 32.4 | 35.5 | 36.2 | 27.7 | 30.7 | **27.0** | 31.3 |
| Order Control | 46.0 | 38.8 | 31.1 | 33.2 | 35.8 | 36.6 | 29.0 | **32.1** | 24.9 | **31.5** |
| Length Control | 45.1 | 39.4 | 33.4 | 32.4 | **39.0** | **37.6** | **31.1** | 31.1 | 26.8 | 29.9 |
| Combined Control | 44.4 | 38.8 | **33.5** | **33.5** | 36.8 | 36.7 | 30.9 | 31.5 | 27.1 | 30.9 |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | **48.2** | 42.2 | 22.2 | **22.5** | **28.4** | **28.6** | 17.6 | 19.1 | **9.3** | **11.2** |
| Order Control | 45.0 | 39.5 | 21.6 | 20.8 | 25.7 | 23.6 | 16.4 | 17.7 | 7.4 | 8.0 |
| Length Control | 47.8 | **42.5** | **22.8** | **22.5** | 27.5 | 26.8 | **18.2** | **19.2** | 9.0 | 10.5 |
| Combined Control | 44.5 | 38.8 | 21.5 | 20.8 | 26.5 | 25.3 | 17.0 | 18.0 | 7.9 | 8.7 |
| **GPT-4o** | | | | | | | | | | |
| Baseline | **56.8** | **49.7** | 26.0 | 28.0 | **33.2** | **34.1** | 20.1 | 24.7 | 11.4 | **14.7** |
| Order Control | 54.5 | 48.3 | 24.2 | 26.5 | 30.0 | 31.8 | 18.5 | 23.4 | 9.6 | 11.9 |
| Length Control | 55.2 | 49.5 | **28.6** | **29.3** | 31.6 | 32.6 | **22.4** | **25.4** | **11.6** | 13.1 |
| Combined Control | 53.3 | 47.7 | 25.8 | 27.1 | 32.7 | 33.5 | 20.7 | 23.9 | 9.8 | 10.8 |
| **Absent Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | **6.8** | **5.5** | 4.6 | **3.8** | **3.2** | **3.0** | 3.8 | 3.0 | **4.6** | 3.6 |
| Order Control | 5.4 | 4.4 | 4.0 | 3.3 | 3.0 | 2.7 | 3.9 | **3.2** | 4.5 | **3.8** |
| Length Control | 5.3 | 4.1 | 4.2 | 3.4 | 2.4 | 2.2 | 3.9 | 3.0 | 4.4 | 3.6 |
| Combined Control | 4.7 | 3.6 | **4.7** | **3.8** | 2.7 | 2.4 | **4.0** | 3.1 | 4.4 | 3.6 |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | **7.3** | **6.3** | 1.3 | 1.1 | **2.0** | 1.5 | **1.3** | **1.1** | **0.4** | **0.4** |
| Order Control | 6.2 | 5.2 | 1.5 | **1.3** | 1.3 | 1.2 | 1.2 | 1.0 | 0.3 | 0.3 |
| Length Control | 6.8 | 5.6 | 1.5 | 1.1 | 1.9 | **1.8** | **1.3** | **1.1** | **0.4** | **0.4** |
| Combined Control | 6.4 | 5.4 | **1.6** | **1.3** | 1.2 | 1.1 | 1.2 | 1.0 | 0.3 | 0.3 |
| **GPT-4o** | | | | | | | | | | |
| Baseline | 10.6 | 10.6 | **4.0** | **4.0** | **2.6** | **2.6** | **2.4** | **2.5** | 0.4 | 0.5 |
| Order Control | 9.8 | 9.5 | 2.6 | 2.6 | 2.3 | 2.2 | 2.0 | 2.1 | 0.4 | 0.5 |
| Length Control | 9.9 | 9.8 | 2.2 | 2.0 | 2.5 | 2.5 | **2.4** | **2.5** | **0.6** | **0.6** |
| Combined Control | **11.0** | **11.0** | 2.5 | 2.5 | 1.8 | 1.8 | 1.9 | 1.9 | 0.3 | 0.3 |

Table 2: Comparison of baseline prompts and prompts with additional instructions for present and absent keyphrase generation.

and datasets. We explain our evaluation in Appendix A.

## 3.1 Datasets

In our experiments we explored a number of datasets that focus on the domain of scientific publications (SemEval (Kim et al., 2010), Krapivin (Krapivin et al., 2009), KP20K (Liu et al., 2020), Inspec (Joshi et al., 2023)), and also a dataset focusing on the news domain (KPTimes (Gallina et al., 2019)). These datasets are commonly used as benchmarks KPG. All experiments were performed in a zero-shot setting solely on the test subsets of the datasets. For SemEval, Krapivin, and Inspec, we utilized the full datasets across all our models: Llama-3, Phi-3, and GPT-4o. For KP20K and KPTimes, we employed the full datasets for Llama-3 and Phi-3, as they are open-source models. However, for the closed-source model GPT-4o, we used a subset of 2,000 samples from each dataset to make the experiments cost-effective. We also show the comparison between LLama-3, Phi-3, and GPT-4o on the same 2,000 samples for KPTimes and KP20K in the Appendix Table 8 and observe similar patterns as in the main paper.

## 3.2 Specialist Prompts Results (RQ1)

In Table 1, we present the results of our baseline and specialist (present and absent) prompts. Interestingly, we find that the specialist present and absent prompt do not consistently outperform the baseline; rather in many cases underperform compared to the baseline both in present and absent keyphrase generation. Interestingly, GPT-4o despite being estimatedly a much larger model still shows no consistent benefit from the specialized prompts; moreover, it also seems to perform worse than LLama-3 for KPG on most datasets. Thus, at least for the explored LLM-based models and the considered prompts, the answer to RQ1 seems to be negative.[4]

sampling. We also did not find substantial differences from different temperatures in a subset of the KP20K validation set.

[4]As would be expected given that the specialists individually do not outperform the baseline, in our experiments, the ensembling of the two specialist models also failed to outperform the ensembling of two baseline prompt-based models.

| Models | Inspec F1@M | Inspec F1@5 | Krapivin F1@M | Krapivin F1@5 | SemEval F1@M | SemEval F1@5 | KP20K F1@M | KP20K F1@5 | KPTimes F1@M | KPTimes F1@5 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Present Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | 48.3 | 40.5 | 30.9 | 32.4 | 35.5 | 36.2 | 27.7 | 30.7 | **27.0** | 31.3 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 36.5 | 30.1 | 22.2 | 18.0 | 26.6 | 21.7 | 18.9 | 16.0 | 13.0 | 9.5 |
| Union Concat | **50.0** | 42.6 | 30.6 | 32.2 | 37.6 | 35.1 | 27.3 | 31.0 | 23.4 | 31.4 |
| Union Interleaf | 42.4 | 36.4 | 30.5 | 32.3 | **38.3** | 36.3 | **29.1** | 31.5 | 25.9 | **32.3** |
| Frequency Order | 49.9 | **45.6** | **31.8** | **33.6** | 38.0 | **38.1** | 28.7 | **32.1** | 24.7 | 31.5 |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | 48.2 | 42.2 | 22.2 | 22.5 | 28.4 | 28.6 | 17.6 | 19.1 | 9.3 | 11.2 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 33.8 | 29.8 | 16.9 | 15.6 | 18.7 | 14.5 | 12.9 | 11.1 | 6.7 | 5.6 |
| Union Concat | 50.2 | 45.5 | 23.1 | 22.7 | 30.4 | 30.3 | 18.0 | 19.8 | 10.9 | 12.2 |
| Union Interleaf | 45.2 | 41.0 | 24.9 | 25.3 | **33.2** | **31.4** | **21.6** | **22.5** | **15.1** | **14.9** |
| Frequency Order | **54.7** | **50.9** | **25.1** | **24.7** | 32.9 | 30.5 | 19.7 | 20.4 | 12.0 | 11.9 |
| **GPT-4o** | | | | | | | | | | |
| Baseline | 56.8 | 49.7 | 26.0 | 28.0 | 33.2 | 34.1 | 20.1 | 24.7 | 11.4 | 14.7 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 46.0 | 36.4 | 21.7 | 18.0 | 24.3 | 17.8 | 15.7 | 13.2 | 8.8 | 7.4 |
| Union Concat | 57.6 | 50.4 | 25.9 | 28.1 | **33.7** | 34.4 | 20.1 | 24.6 | 12.9 | 15.5 |
| Union Interleaf | 54.2 | 47.0 | **27.7** | **29.4** | 33.2 | **34.5** | **21.9** | **26.3** | **16.0** | **18.2** |
| Frequency Order | **58.2** | **52.8** | 25.9 | 25.6 | 32.7 | 31.4 | 20.0 | 21.7 | 12.1 | 12.4 |
| **Absent Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | 6.8 | 5.5 | 4.6 | 3.8 | 3.2 | 3.0 | 3.8 | 3.0 | 4.6 | 3.6 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 3.7 | 4.9 | 3.9 | 3.8 | 1.7 | 1.2 | 2.8 | 3.2 | 2.1 | 2.3 |
| Union Concat | **8.9** | **8.2** | 5.4 | 4.7 | 3.6 | **3.6** | 4.6 | 4.5 | 4.6 | 4.4 |
| Union Interleaf | 6.8 | 6.3 | 5.2 | 4.9 | 3.0 | 2.7 | 5.0 | 4.7 | 4.9 | 4.7 |
| Frequency Order | 8.5 | 7.6 | **5.9** | **5.1** | **3.9** | **3.6** | **5.4** | **4.9** | **5.3** | **5.0** |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | 7.3 | 6.3 | 1.3 | 1.1 | 2.0 | 1.5 | 1.3 | 1.1 | 0.4 | 0.4 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 2.7 | 2.7 | 0.8 | 0.7 | 1.1 | 1.1 | 0.8 | 0.8 | 0.2 | 0.2 |
| Union Concat | 8.2 | 7.8 | 1.8 | 1.8 | 1.9 | 1.8 | 1.5 | 1.5 | 0.4 | 0.5 |
| Union Interleaf | 6.2 | 6.0 | 1.7 | 1.8 | 1.3 | 0.9 | 1.6 | 1.6 | 0.4 | 0.4 |
| Frequency Order | **9.3** | **9.0** | **2.1** | **2.1** | **2.5** | **2.7** | **2.0** | **2.0** | **0.6** | **0.7** |
| **GPT-4o** | | | | | | | | | | |
| Baseline | 10.6 | 10.6 | **4.0** | **4.0** | 2.6 | 2.6 | 2.4 | 2.5 | 0.4 | 0.5 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 5.6 | 6.8 | 1.8 | 1.9 | 1.6 | 2.1 | 1.3 | 1.5 | 0.3 | 0.2 |
| Union Concat | 10.2 | 9.7 | **3.9** | **3.7** | 2.9 | 3.3 | 2.5 | 2.3 | 0.4 | 0.5 |
| Union Interleaf | 10.9 | 10.1 | 3.3 | 3.0 | 1.9 | 2.4 | **2.8** | **2.8** | **0.6** | **0.7** |
| Frequency Order | **11.7** | **10.8** | 3.5 | 3.1 | **3.7** | **3.6** | 2.6 | 2.6 | **0.6** | 0.6 |

Table 3: Comparison of baseline models and multisample models with different aggregation strategies for both present and absent keyphrase generation.

### 3.3 Additional Instruction Results (RQ2)

In Table 2, we present the results of including additional instructions to the baseline prompt for order control and length control as discussed before. Here, we find that Length Control can sometimes help in the performance of present keyphrase extraction in some datasets. However, the overall result is mixed, and none of the strategies of additional instructions consistently improve the baseline across both present and absent keyphrase generation. As such, the answer to RQ2 also seems to lead towards a negative outcome.

### 3.4 Multi-Sampling Results (RQ3)

In Table 3, we present the results of multi-sampling with various aggregation strategies. As we would expect, simple union does not help much, and often harms the performance because it removes all order information (which is relevant). Because of our dynamic keyphrase number selection strategy, the order is relevant even for @M metrics. Union Concat, Union Interleaf, and Frequency Order are the three best contenders for multi-sampling aggregation. Among the three, Frequence Order-based aggregation consistently shows the best performance; particularly, on absent keyphrase generation for the open-source models. Overall, we find that the best aggregation methods with multi-sampling significantly improve the performance of LLMs over the baseline. As such, the answer to RQ3 leans towards an affirmation. In Appendix Table 5, we also show how well absent keyphrases are recalled for the

| Models | Inspec F1@M | Inspec F1@5 | Krapivin F1@M | Krapivin F1@5 | SemEval F1@M | SemEval F1@5 | KP20K F1@M | KP20K F1@5 | KPTimes F1@M | KPTimes F1@5 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Present Keyphrase Generation** | | | | | | | | | | |
| catSeqTG (Chan et al., 2019) | 27.0 | 22.9 | 36.6 | 28.2 | 29.0 | 24.6 | 36.6 | 29.2 | — | — |
| catSeqTG-2RF1 (Chan et al., 2019) | 30.1 | 25.3 | 36.9 | 30.0 | 32.9 | 28.7 | 38.6 | 32.1 | — | — |
| ExHiRD-h (Chen et al., 2020) | $29.1_{3}$ | $25.3_{4}$ | $34.7_{4}$ | $28.6_{4}$ | $33.5_{17}$ | $28.4_{15}$ | $37.4_{0}$ | $31.1_{1}$ | — | — |
| Transformer (Ye et al., 2021b) | $32.5_{6}$ | $28.1_{5}$ | $36.5_{5}$ | $31.5_{8}$ | $32.5_{15}$ | $28.7_{14}$ | $37.7_{1}$ | $33.2_{1}$ | — | — |
| SetTrans (Ye et al., 2021b) * | $32.4_{3}$ | $28.5_{3}$ | $36.4_{12}$ | $32.6_{12}$ | $35.7_{13}$ | $33.1_{20}$ | $39.2_{4}$ | $35.8_{5}$ | 54.8 | — |
| KPD-A (Chowdhury et al., 2022) * | $30.6_{3}$ | $25.7_{3}$ | $35.3_{6}$ | $29.5_{7}$ | $34.4_{5}$ | $30.3_{7}$ | $39.6_{2}$ | $33.9_{3}$ | 55.5 | — |
| Diversity Heads (Thomas and Vajjala, 2024) | 32.1 | — | 37.4 | — | 39.6 | — | 41.7 | — | **56.3** | — |
| UniKeyphrase (Wu et al., 2021) * | 31.1 | 29.0 | — | — | **40.9** | **41.6** | **42.8** | 40.8 | 34.5 | — |
| PromptKP (Wu et al., 2022c) | 29.4 | 26.0 | — | — | 35.6 | 32.9 | 35.5 | 35.1 | — | — |
| SciBART-large (Wu et al., 2023) | 40.2 | — | 35.2 | — | 34.1 | — | 43.1 | — | — | — |
| SimCKP (Choi et al., 2023) | $35.8_{8}$ | $35.6_{6}$ | $\mathbf{40.5_{8}}$ | $\mathbf{40.5_{8}}$ | $38.6_{4}$ | $38.7_{2}$ | $42.7_{1}$ | $\mathbf{42.6_{1}}$ | — | — |
| ChatGPT TP4 (Song et al., 2023b) | 39.3 | 32.2 | 16.3 | 17.0 | 21.2 | 23.3 | 13.6 | 16.0 | — | — |
| **Ours** | | | | | | | | | | |
| Llama-3 Multi-sampling | 49.9 | 45.6 | 31.8 | 33.6 | 38.0 | 38.1 | 28.7 | 32.1 | 24.7 | 31.5 |
| Phi-3 Multi-sampling | 54.7 | 50.9 | 25.1 | 24.7 | 32.9 | 30.5 | 19.7 | 20.4 | 12.0 | 11.9 |
| GPT-4o | **58.2** | **52.8** | 25.9 | 25.6 | 32.7 | 31.4 | 20.0 | 21.7 | 12.1 | 12.4 |
| **Absent Keyphrase Generation** | | | | | | | | | | |
| catSeqTG (Chan et al., 2019) | 1.1 | 0.5 | 3.4 | 1.8 | 2.7 | 1.9 | 3.2 | 1.5 | — | — |
| catSeqTG-2RF1 (Chan et al., 2019) | 2.1 | 1.2 | 5.3 | 3.0 | 3.0 | 2.1 | 5.0 | 2.7 | — | — |
| ExHiRD-h (Chen et al., 2020) | $2.2_{3}$ | $1.1_{1}$ | $4.3_{6}$ | $2.2_{3}$ | $2.5_{6}$ | $1.7_{4}$ | $3.2_{0}$ | $1.6_{0}$ | — | — |
| Transformer (Ye et al., 2021b) | $1.9_{4}$ | $1.0_{2}$ | $6.0_{4}$ | $3.2_{1}$ | $2.3_{3}$ | $2.0_{5}$ | $4.6_{1}$ | $2.3_{1}$ | — | — |
| SetTrans (Ye et al., 2021b) * | $3.4_{3}$ | $2.1_{1}$ | $7.3_{11}$ | $4.7_{7}$ | $3.4_{5}$ | $2.6_{3}$ | $5.8_{3}$ | $3.6_{2}$ | 41.2 | — |
| KPD-A (Chowdhury et al., 2022) * | $3.2_{2}$ | $2.1_{1}$ | $7.2_{7}$ | $4.6_{4}$ | $\mathbf{4.7_{1}}$ | $3.6_{1}$ | $6.6_{1}$ | $4.2_{1}$ | 42.6 | — |
| Diversity Heads (Thomas and Vajjala, 2024) | 1.2 | — | 7.6 | — | 4.2 | — | 7.8 | — | **44.1** | — |
| UniKeyphrase (Wu et al., 2021) * | 2.9 | 2.9 | — | — | 3.2 | 3.0 | 4.7 | 4.7 | 20.8 | — |
| PromptKP (Wu et al., 2022c) | 2.2 | 1.7 | — | — | 3.2 | 2.8 | 4.2 | 3.2 | — | — |
| SciBART-large (Wu et al., 2023) | 3.6 | — | 8.6 | — | 4.0 | — | 7.6 | — | — | — |
| SimCKP (Choi et al., 2023) | $3.5_{3}$ | $3.3_{2}$ | $\mathbf{8.9_{0}}$ | $\mathbf{7.8_{1}}$ | $\mathbf{4.7_{6}}$ | $\mathbf{4.0_{2}}$ | $\mathbf{8.0_{1}}$ | $\mathbf{7.3_{2}}$ | — | — |
| ChatGPT TP4 (Song et al., 2023b) | 4.1 | 3.0 | 1.5 | 1.1 | 0.5 | 0.4 | 3.9 | 3.8 | — | — |
| **Ours** | | | | | | | | | | |
| Llama-3 Multi-sampling | 8.5 | 7.6 | 5.9 | 5.1 | 3.9 | 3.6 | 5.4 | 4.9 | 5.3 | 5.0 |
| Phi-3 Multi-sampling | 9.3 | 9.0 | 2.1 | 2.1 | 2.5 | 2.7 | 2.0 | 2.0 | 0.6 | 0.7 |
| GPT-4o | **11.7** | **10.8** | 3.5 | 3.1 | 3.7 | 3.6 | 2.6 | 2.6 | 0.6 | 0.6 |

Table 4: We compare the performance of our models with various prior works (results from prior works are copied from the corresponding citations; the citations here indicate the source of the results and not necessarily the original work presenting the relevant methods). * Indicates that the kptimes result are taken from (Thomas and Vajjala, 2024) rather than the corresponding citation. Llama3/Phi3/GPT 4o Multisample denotes Llama3/Phi3/GPT 4o multisample (n=10) results with frequency-based ordering and aggregation. KPD-A denotes SetTrans with Greedy Search + KPDrop-A. For brevity, we only present the greedy search results of Diversity Heads (Thomas and Vajjala, 2024) and TP4 prompt style for ChatGPT. SciBART-large indicates the result of (SciBART-large+TAPT+DESEL in Wu et al. (2023)). $91_1$ denotes $91 \pm 0.1$.

multi-sampling-based approaches compared to the baseline.

### 3.5 Comparison with Prior Works

As can be seen in Table 4, our best approaches are competitive against many of the earlier works. Our LLM-based models tend to generate high number of keyphrases which is well suited for Inspec (which also has a high number of keyphrases in the ground truth). As such, our model excels and achieves state of the art in Inspec. In other cases, the overgeneration can become a detriment leading to lower precision when ground truth keyphrases are of fewer numbers. Regardless, our models still remain competitive against many of the prior models in scientific documents. This is especially impressive because this performance is completely zero-shot without any fine-tuning, unlike most prior works. Interestingly, the LLM-based models seem to perform quite poorly in the news domain (KP-Times) compared to others. The gap is particularly high in absent keyphrase generation for KP-Times. Thus, it appears that the LLM-based models, in a zero-shot context, are better biased towards scientific keyphrase extraction, rather than KP-Times-style news domain.

## 4 Additional Analyses

In Appendix Table 5 we show the recall of absent keyphrases at higher top-ks. In Appendix B, we present result of using multi-sampling aggregations on beam search generations as opposed to independent random sampling. As can be seen independent-sampling based multi-sampling generally outperforms beam search while being more cost-efficient. In Appendix C, we provide qualitative analyses of generated keyphrases. In brief, we find that, under zero-shot prompts, models are biased towards producing high number of longer (multi-word) keyphrases. Inspec best fits this pattern, and thus we find LLMs to ace on Inspec. Whereas KPTimes tend to have short keyphrases and in fewer numbers - potentially a reason for the struggle of zero-shot LLMs in KPTimes.

## 5 Related Work

Identifying keyphrase from a document is a longstanding task and has been well studied in the literature using both supervised, semi-supervised, and unsupervised approaches (Patel and Caragea, 2021; Patel et al., 2020; Park and Caragea, 2020; Chowdhury et al., 2019; Patel and Caragea, 2019; Ye and Wang, 2018; Florescu and Caragea, 2017; Hasan and Ng, 2014; Gollapalli and Caragea, 2014; Bougouin et al., 2013; Mihalcea and Tarau, 2004). However, with the surge of deep learning models, the attention has shifted towards generative models particularly because of their capability to generate absent keyphrases (Wu et al., 2024a; Garg et al., 2023, 2022; Chowdhury et al., 2022; Meng et al., 2017). Many recent works for keyphrase generation have also explored the seq2seq models with no pre-training (Meng et al., 2017; Chen et al., 2018; Ye and Wang, 2018; Chan et al., 2019; Swaminathan et al., 2020; Chen et al., 2020; Ye et al., 2021b,a; Huang et al., 2021; Choi et al., 2023; Thomas and Vajjala, 2024) or pre-trained seq2seq models (e.g., BART) for generating both absent and present keyphrases (Liu et al., 2020; Wu et al., 2021; Kulkarni et al., 2022; Wu et al., 2022b,a; Garg et al., 2022; Chowdhury et al., 2022; Madaan et al., 2022; Wu et al., 2023, 2024a).

More recently, a few works have started to explore decoder-only LLMs for keyphrase generation and extraction (Wang et al., 2024; Maragheh et al., 2023; Song et al., 2023a,b; Martínez-Cruz et al., 2023; Wu et al., 2024b). In our paper, we explore LLMs using novel strategies such as "specialist prompts", task-specific instructions, and multi-sampling, and contrast them with many of the above works.

## 6 Conclusion and Future Work

In this paper, we addressed three core research questions for keyphrase generation: the effectiveness of specialist prompting for present and absent keyphrases (RQ1), the impact of additional instructions for length and order control (RQ2), and the benefits of multi-sampling for improving keyphrase generation (RQ3). For RQ1, we found that the specialist prompts for present and absent keyphrases did not consistently outperform a simple baseline prompt. In terms of RQ2, introducing additional instructions for order and length control yielded mixed results. While length control showed some promise in improving present keyphrase extraction for specific datasets, the overall performance gains were inconsistent across both present and absent keyphrase generation. The most promising findings of our paper came from our exploration of RQ3 — the impact of multi-sampling and aggregation. Simple union proved insufficient due to its inability to preserve keyphrase order, which is crucial for certain evaluation metrics like $F_1$@5. However, more sophisticated aggregation techniques, such as Union Concatenation, Union Interleaf, and especially Frequency Order, showed significant improvements in keyphrase generation, particularly for absent keyphrases. Frequency Order, in particular, provided the most consistent results and outperformed the baseline across various settings.

Our multi-sampling aggregation strategies are also model-agnostic and can work with earlier established KPG models. We leave potential to augment earlier model strategies with multi-sampling aggregation for future work.

## 7 Limitations

This work focuses on zero-shot prompting; however, the effectiveness of few-shot prompting, and parameter-efficient fine-tuning for KPG are also relevant questions that are yet unanswered in this paper. Moreover, alternative evaluation schemes to better judge LLM's capacities such as KPEval (Wu et al., 2024b) are yet to be explored. Despite these limitations, we believe our LLM-based methods show promise and offer a strong foundation for future work in LLM-based keyphrase generation.

## Acknowledgements

This research is funded in part by NSF CAREER award 1802358, NSF CRI award 1823292, NSF IIS award 2107518, and an award from Discovery Partners Institute (DPI) at the University of Illinois Chicago. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF or DPI.

## References


Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *ACL*, pages 500–509. ACL.

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*.

Rabah A. Al-Zaidy, Cornelia Caragea, and C. Lee Giles. 2019. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2551–2557. ACM.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229, Brussels, Belgium. Association for Computational Linguistics.

Florian Boudin, Ygor Gallina, and Akiko Aizawa. 2020. Keyphrase generation for scientific document retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1126.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 543–551. Asian Federation of Natural Language Processing / ACL.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174, Florence, Italy. Association for Computational Linguistics.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *EMNLP*, pages 4057–4066.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105, Online. Association for Computational Linguistics.

Minseok Choi, Chaeheon Gwak, Seho Kim, Si Kim, and Jaegul Choo. 2023. SimCKP: Simple contrastive learning of keyphrase representations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3003–3015, Singapore. Association for Computational Linguistics.

Jishnu Ray Chowdhury, Cornelia Caragea, and Doina Caragea. 2019. Keyphrase extraction from disaster-related tweets. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1555–1566. ACM.

Jishnu Ray Chowdhury, Seoyeon Park, Tuhin Kundu, and Cornelia Caragea. 2022. KPDROP: improving absent keyphrase generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4853–4870. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1105–1115. Association for Computational Linguistics.

Ygor Gallina, Florian Boudin, and Beatrice Daille. 2019. Kptimes: A large-scale dataset for keyphrase generation on news documents. *arXiv preprint arXiv:1911.12559*.

Krishna Garg, Jishnu Ray Chowdhury, and Cornelia Caragea. 2023. Data augmentation for low-resource keyphrase generation. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8442–8455. Association for Computational Linguistics.

Krishna Garg, Jishnu Ray Chowdhury, and Cornelia Caragea. 2022. Keyphrase generation beyond the boundaries of title and abstract. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5809–5821, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27*

-31, 2014, Québec City, Québec, Canada, pages 1629–1635. AAAI Press.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1262–1273. The Association for Computer Linguistics.

Xiaoli Huang, Tongge Xu, Lvan Jiao, Yueran Zu, and Youmin Zhang. 2021. Adaptive beam search decoding for discrete keyphrase generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):13082–13089.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*.

Steve Jones and Mark S Staveley. 1999. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167.

Rishabh Joshi, Vidhisha Balachandran, Emily Saldanha, Maria Glenski, Svitlana Volkova, and Yulia Tsvetkov. 2023. Unsupervised keyphrase extraction via interpretable neural networks. *Preprint*, arXiv:2203.07640.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction.

Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2022. Learning rich representation of keyphrases from text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 891–906, Seattle, United States. Association for Computational Linguistics.

Rui Liu, Zheng Lin, and Weiping Wang. 2020. Keyphrase prediction with pre-trained language model. *CoRR*, abs/2004.10462.

Aman Madaan, Dheeraj Rajagopal, Niket Tandon, Yiming Yang, and Antoine Bosselut. 2022. Conditional set generation using seq2seq models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4874–4896, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Reza Yousefi Maragheh, Chenhao Fang, Charan Chand Irugu, Parth Parikh, Jason Cho, Jianpeng Xu, Saranyan Sukumar, Malay Patel, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2023. Llm-take: Theme aware keyword extraction using large language models. *Preprint*, arXiv:2312.00909.

Roberto Martínez-Cruz, Alvaro J López-López, and José Portela. 2023. Chatgpt vs state-of-the-art models: a benchmarking study in keyphrase generation task. *arXiv preprint arXiv:2304.14177*.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 404–411. ACL.

Pierre Moulin and Mehmet Kivanç Mihcak. 2002. A framework for evaluating the data-hiding capacity of image sources. *IEEE Transactions on Image Processing*, 11(9):1029–1042.

NBC News. 2019. Manafort family business defends its name as infamous cousin sits in jail. Accessed: 2025-01-26.

Seo Park and Cornelia Caragea. 2023. Multi-task knowledge distillation with embedding constraints for scholarly keyphrase boundary classification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 13026–13042. Association for Computational Linguistics.

Seoyeon Park and Cornelia Caragea. 2020. Scientific keyphrase identification and classification by pre-trained language models intermediate task transfer learning. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 5409–5419. International Committee on Computational Linguistics.

Krutarth Patel and Cornelia Caragea. 2019. Exploring word embeddings in crf-based keyphrase extraction from research papers. In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019*, pages 37–44. ACM.

Krutarth Patel and Cornelia Caragea. 2021. Exploiting position and contextual word embeddings for keyphrase extraction from scientific papers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19*

- *23, 2021*, pages 1585–1591. Association for Computational Linguistics.

Krutarth Patel, Cornelia Caragea, Jian Wu, and C. Lee Giles. 2020. Keyphrase extraction in scholarly digital library search engines. In *Web Services - ICWS 2020 - 27th International Conference, Held as Part of the Services Conference Federation, SCF 2020, Honolulu, HI, USA, September 18-20, 2020, Proceedings*, volume 12406 of *Lecture Notes in Computer Science*, pages 179–196. Springer.

Ulf-Dietrich Reips and Christoph Neuhaus. 2002. Wextor: A web-based tool for generating and visualizing experimental designs and procedures. *Behavior Research Methods, Instruments, & Computers*, 34:234–240.

Min Song, Il Yeol Song, Robert B. Allen, and Zoran Obradovic. 2006. Keyphrase extraction-based query expansion in digital libraries. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '06, page 202–209, New York, NY, USA. Association for Computing Machinery.

Mingyang Song, Xuelian Geng, Songfang Yao, Shilong Lu, Yi Feng, and Liping Jing. 2023a. Large language models as zero-shot keyphrase extractor: A preliminary empirical study. *arXiv preprint arXiv:2312.15156*.

Mingyang Song, Haiyun Jiang, Shuming Shi, Songfang Yao, Shilong Lu, Yi Feng, Huafeng Liu, and Liping Jing. 2023b. Is chatgpt a good keyphrase generator? a preliminary study. *arXiv preprint arXiv:2303.13001*.

Lucas Sterckx, Cornelia Caragea, Thomas Demeester, and Chris Develder. 2016. Supervised keyphrase extraction as positive unlabeled learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1924–1929. The Association for Computational Linguistics.

Avinash Swaminathan, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, Rajiv Ratn Shah, and Amanda Stent. 2020. A preliminary exploration of GANs for keyphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8021–8030, Online. Association for Computational Linguistics.

Edwin Thomas and Sowmya Vajjala. 2024. Improving absent keyphrase generation with diversity heads. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1568–1584, Mexico City, Mexico. Association for Computational Linguistics.

The Japan Times. 2014. Chinese tourists step up as abe, japanese tighten belts. Accessed: 2025-01-26.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yang Wang, Zheyi Sha, Kunhai Lin, Chaobing Feng, Kunhong Zhu, Lipeng Wang, Xuewu Jiao, Fei Huang, Chao Ye, Dengwu He, Zhi Guo, Shuanglong Li, and Lin Liu. 2024. One-step reach: Llm-based keyword generation for sponsored search advertising. In *Companion Proceedings of the ACM on Web Conference 2024*, WWW '24, page 1604–1608, New York, NY, USA. Association for Computing Machinery.

Di Wu, Wasi Ahmad, and Kai-Wei Chang. 2023. Rethinking model selection and decoding for keyphrase generation with pre-trained sequence-to-sequence models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6642–6658, Singapore. Association for Computational Linguistics.

Di Wu, Wasi Ahmad, and Kai-Wei Chang. 2024a. On leveraging encoder-only pre-trained language models for effective keyphrase generation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12370–12384, Torino, Italia. ELRA and ICCL.

Di Wu, Wasi Uddin Ahmad, and Kai-Wei Chang. 2022a. Pre-trained language models for keyphrase generation: A thorough empirical study. *arXiv preprint arXiv:2212.10233*.

Di Wu, Wasi Uddin Ahmad, Sunipa Dev, and Kai-Wei Chang. 2022b. Representation learning for resource-constrained keyphrase generation. *ArXiv*, abs/2203.08118.

Di Wu, Da Yin, and Kai-Wei Chang. 2024b. KPEval: Towards fine-grained semantic-based keyphrase evaluation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1959–1981, Bangkok, Thailand. Association for Computational Linguistics.

Huanqin Wu, Wei Liu, Lei Li, Dan Nie, Tao Chen, Feng Zhang, and Di Wang. 2021. UniKeyphrase: A unified extraction and generation framework for keyphrase prediction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 825–835, Online. Association for Computational Linguistics.

Huanqin Wu, Baijiaxin Ma, Wei Liu, Tao Chen, and Dan Nie. 2022c. Fast and constrained absent keyphrase generation by prompt-based learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11495–11503.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4142–4153, Brussels, Belgium. Association for Computational Linguistics.

Jiacheng Ye, Ruijian Cai, Tao Gui, and Qi Zhang. 2021a. Heterogeneous graph neural networks for keyphrase generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2705–2715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021b. One2Set: Generating diverse keyphrases as a set. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4598–4608, Online. Association for Computational Linguistics.

Yang Yu and Vincent Ng. 2018. Improving unsupervised keyphrase extraction using background knowledge. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975.

# A Evaluation

We consider the following standard evaluations for KPG:

1. $F_1$@M: This evaluation metric calculates the $F_1$ between all the predicted keyphrases by the model and the ground truth keyphrases. In the case of multi-sampling models, the term "all the predicted keyphrases" stands for all the keyphrases that remain after selection of the top-$M_{pre}$ present keyphrases and top-$M_{abs}$ absent keyphrases based on the dynamic keyphrase number selection that we discussed before.

2. $F_1$@5: This evaluation metric calculates the $F_1$ between the top-5 predicted keyphrases by the model and the ground truth keyphrases. Similar to (Chan et al., 2019) and others, in case there are less than 5 predicted keyphrases, we add dummy ones until there are 5 keyphrases.

3. R@10: This evaluation metric calculates the recall between the top-10 predicted keyphrases by the model and the ground truth keyphrases.

4. R@Inf: This evaluation metric calculates the recall between all the predicted keyphrases (with no truncation) by the model and the ground truth keyphrases. The difference between @Inf and @M is that in the context of multi-sampling models, for @Inf, we do not truncate the keyphrases based on dynamically determined $M_{pre}$ and $M_{abs}$ values. Otherwise, for any other case, @M and @Inf are equivalent. R@Inf shows the upperbound performance that we can get if we have a perfect selector to select from the raw list of predictions from all samples of a model for any specific input.

For all cases, we calculate the macro-average as is the standard. Following convention, we distinguish between absent and present keyphrases based on whether the lower-cased stemmed (using Porter-Stemmer) version of keyphrases match with the lowercased stemmed version of the input text.

# B Beam Search

Beam search is a search algorithm used in sequence generation tasks, aiming to balance between exploration and exploitation. It maintains a set of the $k$ most probable hypotheses at each step, where $k$ is the beam width. The model computes a probability distribution over the next token, and at each step, the $k$ most probable sequences are kept and expanded. Mathematically, given the sequence $\mathbf{X}_{t-1} = (x_1, x_2, \ldots, x_{t-1})$, the probability of the next token is computed as:

$$P(x_t|\mathbf{X}_{t-1})$$

Beam search proceeds by maintaining and expanding the top $k$ sequences, based on their cumulative probability:

$$P(\mathbf{x}_t) = \prod_{i=1}^{t} P(x_i|\mathbf{X}_{i-1})$$

After expanding all sequences, the top $k$ sequences are retained, and this process repeats until a stopping criterion (e.g., reaching the end token) is met. Following the multi-sampling experiments detailed in Section 3.4, we applied the same aggregation strategies to evaluate the performance of the beam search strategy. Table 7 summarizes the results of beam search conducted on the open-source models Llama-3.0 and Phi-3.0, using a beam width of 10. Consistent with our previous experiments, the generation length was constrained to 500 tokens, with all other parameters held constant. The results indicate that the multi-sampling strategy with various aggregation techniques, consistently outperforms the standard beam search approach across all datasets.

# C Qualitative Analysis of Keyphrase Generation

In the results presented in the main paper, we find that the LLMs perform quite well in Inspec, quite poorly in KPTimes, and moderately competitively in the other datasets. Our analyses, here, provide some insights about why this happens. As the anecdotal examples in Table 9 and Table 10 show, the LLMs (particularly GPT-4o) are biased towards generating high number of keyphrases ($\sim 10$) and more of multi-word keyphrases. Moreover, they are biased towards generating more present keyphrases than absent. This pattern of generation matches very well with the pattern of annotated keyphrases in Inspec (larger number of keyphrases and bigger multi-word keyphrases). On the other hand, the annotated keyphrases in KPTimes are on the opposite side of the spectrum. They have fewer keyphrases

| Models | Inspec | | Krapivin | | SemEval | | KP20K | | KPTimes | |
|---|---|---|---|---|---|---|---|---|---|---|
| | R@10 | R@Inf | R@10 | R@Inf | R@10 | R@Inf | R@10 | R@Inf | R@10 | R@Inf |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | 7.6 | 7.6 | 5.5 | 5.5 | 2.5 | 2.5 | 4.2 | 4.2 | 4.9 | 4.9 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 12.2 | **17.1** | 7.6 | **11.8** | 3.1 | **5.0** | 8.8 | **11.6** | 7.0 | **14.0** |
| Union Concat | 15.6 | **17.1** | 9.6 | **11.8** | 3.5 | **5.0** | 9.8 | **11.6** | 10.0 | **14.0** |
| Union Interleaf | 14.5 | **17.1** | **9.8** | **11.8** | **3.6** | **5.0** | 10.1 | **11.6** | 10.8 | **14.0** |
| Frequency Order | **15.2** | **17.1** | 9.6 | **11.8** | 3.5 | **5.0** | **10.2** | **11.6** | **10.9** | **14.0** |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | 9.4 | 9.4 | 1.7 | 1.7 | 1.9 | 1.9 | 1.8 | 1.9 | 0.7 | 0.8 |
| Multi-sampling (n=10) | | | | | | | | | | |
| Union | 7.5 | **23.8** | 1.8 | **6.9** | 1.7 | **4.6** | 2.5 | **7.5** | 0.7 | **5.2** |
| Union Concat | 16.2 | **23.8** | 3.4 | **6.9** | 2.8 | **4.6** | 3.8 | **7.5** | 1.2 | **5.2** |
| Union Interleaf | 15.4 | **23.8** | 3.4 | **6.9** | 1.6 | **4.6** | 4.2 | **7.5** | 1.3 | **5.2** |
| Frequency Order | **19.7** | **23.8** | **3.8** | **6.9** | **3.0** | **4.6** | **4.6** | **7.5** | **1.8** | **5.2** |

Table 5: Recall performance of our multi-sample models for absent keyphrase generation. R indicates recall. @Inf indicates that all keyphrases from all samples for an input is considered without any dynamic @M selection.

| Dataset | Statistics Names | Original Data (Ground Truth) | Statistics of Model Generations | | |
|---|---|---|---|---|---|
| | | | Llama-3 | Gpt-4o | Phi-3 |
| Inspec | Average words in Title + Abstract | 121.82 | | | |
| | Average words per present keyphrase | 2.27 | 2.00 | 2.36 | 2.43 |
| | Average words per absent keyphrase | 2.52 | 2.14 | 2.69 | 3.07 |
| | Average no. of present keyphrases per input | 7.70 | 7.91 | 8.33 | 7.33 |
| | Average no. of absent keyphrases per input | 2.15 | 2.43 | 3.7 | 5.26 |
| Krapivin | Average Words in Title + Abstract | 180.65 | | | |
| | Average words per present keyphrase | 2.15 | 2.10 | 2.46 | 2.49 |
| | Average words per absent keyphrase | 2.29 | 2.14 | 2.62 | 2.91 |
| | Average no. of present keyphrases per input | 3.28 | 8.75 | 9.63 | 8.29 |
| | Average no. of absent keyphrases per input | 2.57 | 2.75 | 3.46 | 5.57 |
| Semeval | Average words in Title + Abstract | 183.48 | | | |
| | Average words per present keyphrase | 1.91 | 2.02 | 2.29 | 2.34 |
| | Average words per absent keyphrase | 2.22 | 2.08 | 2.54 | 3.42 |
| | Average no. of present keyphrases per input | 6.01 | 9.83 | 8.44 | 7.94 |
| | Average no. of absent keyphrases per input | 8.53 | 3.71 | 3.05 | 5.96 |
| KP20K | Average words in Title + Abstract | 157.94 | | | |
| | Average words per present keyphrase | 1.76 | 2.07 | 2.37 | 2.46 |
| | Average words per absent keyphrase | 2.24 | 2.18 | 2.64 | 3.27 |
| | Average no. of present keyphrases per input | 3.28 | 9.03 | 10.11 | 8.76 |
| | Average no. of absent keyphrases per input | 2.01 | 2.41 | 3.54 | 5.79 |
| KPTimes | Average words in Title + Abstract | 643.24 | | | |
| | Average words per present keyphrase | 1.48 | 1.75 | 2.44 | 2.23 |
| | Average words per absent keyphrase | 2.36 | 2.05 | 3.01 | 2.83 |
| | Average no. of present keyphrases per input | 3.18 | 9.84 | 9.95 | 9.82 |
| | Average no. of absent keyphrases per input | 1.92 | 2.27 | 8.4 | 6.74 |

Table 6: Statistics of datasets and the model generations.

compared to other datasets, and short (typically single word) keyphrases. The statistics of other datasets are in the middle of the spectrum. These trends that we observe in a few anecdotal examples, are also backed quantitatively in Table 6. The table shows several statistics like average number of words per present or absent keyphrases and average number of present and absent keyphrases per input both in model generations and the dataset ground truths. As can be seen, the statistics of model generations correspond most closely to Inspec and least closely to KPTimes. Moreover, the higher average input size of KPTimes may also make things harder for the LLMs. SemEval also has ground truths with higher number of keyphrases comparable to Inspec, but it has much higher ratios of absent keyphrases which conflicts with the pattern of model generations.

All these points can provide a few insights as to why the LLMs perform best in Inspec, worst in KPTimes and neither very good nor very bad in the other datasets.

| Models | Inspec | | Krapivin | | SemEval | | KP20K | | KPTimes | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 |
| **Present Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | **48.3** | 40.5 | 30.9 | **32.4** | 35.5 | 36.2 | 27.7 | **30.7** | 27.0 | **31.3** |
| Beam Search (Beam width=10) | | | | | | | | | | |
| Union | 38.1 | 46.7 | 24.2 | 27.3 | 27.3 | 33.1 | 20.3 | 23.5 | 14.9 | 18.8 |
| Union Concat | 44.4 | **52.0** | **32.5** | 30.7 | **36.4** | 37.0 | 30.7 | 27.2 | 31.2 | 22.9 |
| Union Interleaf | 41.5 | 49.4 | **32.5** | 31.1 | **36.4** | **37.2** | **31.1** | 28.0 | **31.7** | 23.6 |
| Frequency Order | 46.3 | 52.2 | 31.7 | 31.1 | 34.8 | 37.0 | 29.5 | 27.4 | 25.0 | 22.9 |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | **48.2** | 42.2 | 22.2 | 22.5 | 28.4 | 28.6 | 17.6 | **19.1** | 9.3 | **11.2** |
| Beam Search (Beam width=10) | | | | | | | | | | |
| Union | 36.5 | 46.4 | 16.7 | 20.4 | 17.3 | 24.9 | 12.5 | 14.8 | 4.8 | 6.4 |
| Union Concat | 45.1 | 52.1 | 22.1 | 22.2 | 28.9 | 28.9 | 19.0 | 16.7 | 10.3 | 7.3 |
| Union Interleaf | 44.4 | 51.0 | **22.5** | **22.6** | **29.8** | **29.7** | **19.7** | 17.5 | **10.4** | 7.5 |
| Frequency Order | 45.0 | **52.4** | 21.7 | 22.1 | 25.2 | 28.7 | 16.6 | 16.7 | 5.8 | 7.2 |
| **Absent Keyphrase Generation** | | | | | | | | | | |
| **Llama-3.0 8B Instruct** | | | | | | | | | | |
| Baseline | 6.8 | 5.5 | 4.6 | 3.8 | **3.2** | 3.0 | 3.8 | 3.0 | 4.6 | 3.6 |
| Beam Search (Beam width=10) | | | | | | | | | | |
| Union | 7.9 | 6.2 | 4.3 | 3.9 | 2.7 | 2.9 | 4.4 | 3.8 | 4.4 | 3.9 |
| Union Concat | 8.1 | **8.0** | 4.8 | 4.5 | 2.8 | **3.2** | **4.5** | 4.3 | **4.7** | 4.7 |
| Union Interleaf | 8.1 | 6.9 | 4.7 | 4.3 | 2.6 | 2.2 | **4.5** | **4.4** | **4.7** | **4.8** |
| Frequency Order | **8.2** | 7.7 | **4.9** | **4.6** | 2.8 | **3.2** | **4.5** | 4.3 | **4.7** | **4.8** |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | | | |
| Baseline | 7.3 | 6.3 | 1.3 | 1.1 | **2.0** | **1.5** | 1.3 | 1.1 | 0.4 | 0.4 |
| Beam Search (Beam width=10) | | | | | | | | | | |
| Union | 8.3 | 7.8 | 1.6 | 1.3 | 0.8 | 0.7 | 1.4 | 1.2 | 0.4 | 0.3 |
| Union Concat | 9.0 | **9.5** | **1.8** | 1.5 | 1.4 | 1.4 | **1.5** | **1.5** | 0.4 | 0.4 |
| Union Interleaf | 9.0 | 8.5 | 1.7 | **1.6** | 1.4 | **1.5** | **1.5** | **1.5** | 0.4 | 0.4 |
| Frequency Order | **9.3** | **9.5** | 1.7 | 1.5 | 1.4 | 1.3 | **1.5** | **1.5** | **0.5** | **0.5** |

Table 7: Comparison of baseline models and beam search models with different aggregation strategies for both present and absent keyphrase generation.

| Models | Present Keyphrase Generation | | | | Absent Keyphrase Generation | | | |
|---|---|---|---|---|---|---|---|---|
| | KP20K | | KPTimes | | KP20K | | KPTimes | |
| | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 |
| **Llama-3.0 8B Instruct** | | | | | | | | |
| Baseline | 26.8 | 30.0 | **28.3** | 33.0 | 3.2 | 3.0 | **3.9** | **3.8** |
| Union | 18.0 | 15.1 | 13.2 | 9.4 | 2.7 | 3.3 | 1.3 | 1.6 |
| Union Concat | 26.8 | 30.3 | 26.1 | 33.2 | 4.8 | 4.5 | 3.1 | 3.0 |
| Union Interleaf | **28.1** | 30.4 | 28.2 | **33.4** | 5.0 | 4.7 | 3.1 | 3.0 |
| Frequency Order | 27.4 | **30.6** | 26.3 | 31.9 | **5.5** | **4.9** | 3.5 | 3.2 |
| **Phi-3.0 3.8B Mini 128K Instruct** | | | | | | | | |
| Baseline | 17.2 | 19.2 | 9.7 | 11.6 | 1.2 | 1.2 | **0.4** | **0.4** |
| Union | 12.3 | 10.6 | 6.2 | 4.8 | 0.7 | 0.7 | 0.2 | 0.2 |
| Union Concat | 18.8 | 20.6 | 11.7 | 12.5 | 1.6 | 1.7 | 0.3 | **0.4** |
| Union Interleaf | **21.0** | **22.0** | **15.8** | **14.9** | 1.6 | 1.5 | 0.3 | **0.4** |
| Frequency Order | 19.2 | 19.6 | 11.0 | 10.5 | **1.9** | **2.1** | **0.4** | **0.4** |
| **GPT-4o** | | | | | | | | |
| Baseline | 20.1 | 24.7 | 11.4 | 14.7 | 2.4 | 2.5 | 0.4 | 0.5 |
| Union | 15.7 | 13.2 | 8.8 | 7.4 | 1.3 | 1.5 | 0.3 | 0.2 |
| Union Concat | 20.1 | 24.6 | 12.9 | 15.5 | 2.5 | 2.3 | 0.4 | 0.5 |
| Union Interleaf | **21.9** | **26.3** | **16.0** | **18.2** | **2.8** | **2.8** | **0.6** | **0.7** |
| Frequency Order | 20.0 | 21.7 | 12.1 | 12.4 | 2.6 | 2.6 | **0.6** | 0.6 |

Table 8: Comparison of baseline models and multi-sampling models on a subsample of 2,000, using different aggregation strategies for both present and absent keyphrase generation.

| Dataset | Inspec | Krapivin | SemEval | KP20K | KPTimes |
|---|---|---|---|---|---|
| **Title** | Loudspeaker Voice-Coil Inductance Losses: Circuit Models, Parameter Estimation, and Effect on Frequency Response | computation in networks of passively mobile finite state sensors | Computing the Banzhaf Power Index in Network Flow Games | A Graph Coloring Based TDMA Scheduling Algorithm for Wireless Sensor Networks. | Auto sales slide 7.6% in May on minicar tax. |
| **Abstract** | When the series resistance is separated and treated as a separate element, it is shown that losses in an inductor require the ratio of the flux to MMF in the core to be frequency dependent. For small-signal operation, this dependence leads to a circuit model composed of a lossless inductor and a resistor in parallel, both of which are frequency dependent. Mathematical expressions for these elements are derived under the assumption that the ratio of core flux to MMF varies as $\omega^{n-1}$, where n is a constant. A linear regression technique is described for extracting the model parameters from measured data. Experimental data are presented to justify the model for the lossy inductance of a loudspeaker voice-coil. A SPICE example is presented to illustrate the effects of voice-coil inductor losses on the frequency response of a typical driver | we explore the computational power of networks of small resource limited mobile agents . we define two new models of computation based on pairwise interactions of finite state agents in populations of finite but unbounded size . with a fairness condition on interactions , we define the concept of stable computation of a function or predicate , and give protocols that stably compute functions in a class including boolean combinations of threshold k , parity , majority , and simple arithmetic . we prove that all stably computable predicates are in nl . with uniform random sampling of pairs to interact , we define the model of conjugating automata and show that any counter machine with o (n) counters of capacity o ( n ) can be simulated with high probability by a protocol in a population of size n ... | Preference aggregation is used in a variety of multiagent applications, and as a result, voting theory has become an important topic in multiagent system research. However, power indices (which reflect how much real power a voter has in a weighted voting system) have received relatively little attention, although they have long been studied in political science and economics. The Banzhaf power index is one of the most popular; it is also well-defined for any simple coalitional game. In this paper, we examine the computational complexity of calculating the Banzhaf power index within a particular multiagent domain, a network flow game. Agents control the edges of a graph; a coalition wins if it can send a flow of a given size from a source vertex to a target vertex. The relative power of each edge/agent reflects its significance in enabling such a flow, and in real-world networks could be used, for example, to allocate resources for maintaining parts of the network... | Wireless sensor networks should provide with valuable service, which is called service-oriented requirement. To meet this need, a novel distributed graph coloring based time division multiple access scheduling algorithm (GCSA), considering real-time performance for clustering-based sensor network, is proposed in this paper, to determine the smallest length of conflict-free assignment of timeslots for intra-cluster transmissions. GCSA involves two phases. In coloring phase, networks are modeled using graph theory, and a distributed vertex coloring algorithm, which is a distance-2 coloring algorithm and can get colors near to $\delta + 1$, is proposed to assign a color to each node in the network. Then, in scheduling phase, each independent set is mapped to a unique timeslot according to the sets priority which is obtained by considering network structure... | Auto sales in May fell 7.6 percent to 335,644 units from a year ago as the April tax hike on minivehicles weighed on demand, industry bodies said Monday. Minicar sales sank 19.6 percent to 125,755 units, down for the fifth consecutive month, the Japan Light Motor Vehicle and Motorcycle Association said. Minivehicles, which have engine displacements no larger than 660cc, account for around 40 percent of new car sales in Japan. Sales of other cars meanwhile rose 1.4 percent to 209,889 units, rising for the second consecutive month, the Japan Automobile Dealers Association said, hinting the impact of the April 2014 consumption tax hike is on the wane. Demand for cars was sluggish during the fiscal year ended March 31 after the first stage of the doubling of the consumption tax raised the levy by 3 points to 8 percent, tipping Japan into yet another recession. It was the nation's first tax hike in 17 years. The second stage, which has been delayed, will raise it to 10 percent. |
| **Ground Truth Keyphrases** | ["loudspeaker voice-coil inductance losses", "circuit models", "parameter estimation", "frequency response", "series resistance", "small-signal operation", "lossless inductor", "linear regression", "lossy inductance", "SPICE", "loudspeaker driver", "core flux to MMF ratio"] | ["mobile agent", "finite state agent", "stable computation", "sensor net", "diffuse computation", "intermittent communication"] | ["banzhaf power index", "power index", "network flow game", "prefer aggreg", "multiag applic", "vote theori", "vote", "comput complex", "algorithm and problem complex analysi", "social choic theori", "autom agent vote", "probabilist model", "connect game"] | ["graph coloring", "tdma", "distributed", "real-time", "clustering"] | ["autos", "minicars", "car sales"] |
| **Llama-3** | [‘Lossy inductor", ‘Loudspeaker voice-coil", ‘Inductance losses", ‘Circuit modeling", ‘Frequency response", ‘SPICE", ‘Magnetic core", ‘MMF", ‘Flux", ‘Linear regression", ‘Parameter estimation"] | [‘finite state sensors", ‘passive mobile agents", ‘computational power", ‘networks", ‘pairwise interactions", ‘stable computation", ‘boolean combinations", ‘threshold k", ‘parity", ‘majority", ‘arithmetic", ‘NL", ‘uniform random sampling", ‘conjugating automata", ‘counter machine", ‘PRAL", ‘open problems", ‘future directions"] | [‘Computing the Banzhaf Power Index", ‘Network Flow Games", ‘Multiagent System", ‘Power Indices", ‘Preference Aggregation", ‘Voting Theory", ‘Simple Coalitional Games", ‘Graph", ‘Network Flow", ‘Coalition", ‘Resource Allocation", ‘Computational Complexity", ‘#P-completeness", ‘Polynomial Algorithm"] | ["Graph Coloring", "TDMA Scheduling", "Wireless Sensor Networks", "Service-Oriented", "Distributed Algorithm", "Real-Time Performance", "Conflict-Free Assignment", "Time Division Multiple Access", "Clustering-Based Sensor Network", "Graph Theory", "Distributed Vertex Coloring Algorithm", "Distance-2 Coloring", "Network Structure", "Intra-Cluster Delay", "Intra-Cluster Throughput", "Communication Reliability"] | ["Auto sales", "minicar tax", "Japan", "April tax hike", "minivehicles", "car sales", "consumption tax"] |
| **Phi-3** | ["Lossy loudspeaker voice-coil", "Inductor inductance losses", "Frequency dependent losses", "Lossless inductor model", "Resistor model", "Small-signal operation", " Mathematical expressions derivation", "Core flux to MMF ratio", "Linear regression technique", "Loudspeaker frequency response", "SPICE example"] | ["small resource limited mobile agents", "computation in networks", "finite state agents", "stable computation", "boolean combinations", "conjugating automata", "counter machine simulation", "high probability", "promising future directions"] | ["Banzhaf power index", "network flow game", "computational complexity", "#P-complete", "polynomial algorithm", "coalitional game", "voting theory", "resource allocation", "real-world networks", "graph edge significance"] | ["Graph Coloring", "Time Division Multiple Access - (TDMA) Scheduling", "Service-Oriented Requirement", "Clustering-Based Sensor Network", "Distributed Vertex Coloring Algorithm", "Distance-2 Coloring Algorithm", "Intra-Cluster Transmissions", "Intra-Cluster Delay", "Intra-Cluster Throughput", "Real-Time Performance", "Communication Reliability", "Experimental Results"] | ["auto sales decline", "minicar tax impact", "minicar sales drop", "consumption tax hike", "car demand slump", "Japan"s recession", "tax policy change"] |
| **GPT-4o** | ["loudspeaker voice-coil inductance", "circuit models", "parameter estimation", "frequency response", "inductor losses", "series resistance", "small-signal operation", "lossless inductor", "resistor in parallel", "frequency dependent elements", "core flux to MMF ratio", "mathematical expressions", "linear regression technique", "lossy inductance", "SPICE example", "loudspeaker driver"] | ["passively mobile finite state sensors", "computational power of networks", "small resource limited mobile agents", "pairwise interactions", "stable computation", "boolean combinations", "threshold functions", "parity", "majority", "simple arithmetic", "stably computable predicates", "NL complexity class", "uniform random sampling", "conjugating automata", "counter machine simulation", "high probability computational model", "PRL complexity class", "open problems", "future directions"] | ["Banzhaf power index", "network flow games", "multiagent systems", "computational complexity", "power indices", "weighted voting system", "coalitional game", "network flow", "#P-complete", "polynomial algorithm", "preference aggregation", "voting theories", "resource allocation", "multiagent domain", "graph edges control"] | ["graph coloring", "TDMA scheduling", "wireless sensor networks", "distributed algorithm", "time division access", "intra-cluster transmissions", "clustering-based sensor network", "real-time performance", "conflict-free assignment", "vertex coloring algorithm", "independent set", "intra-cluster delay", "intra-cluster throughput", "communication reliability"] | ["auto sales decline", "minicar tax impact", "Japan auto market", "minivehicle sales", "consumption tax hike", "Japan recession", "minicar sales decrease", "tax impact on demand", "car sales increase", "Japan Light Motor Vehicle and Motorcycle Association", "Japan Automobile Dealers Association", "fiscal year car sales", "tax hike effects"] |

Table 9: An example of input document from each dataset, along with the corresponding keyphrase generations by various models, where blue represents present keyphrases and black represents absent keyphrases.

| Dataset | Inspec | | KPTimes | |
|---|---|---|---|---|
| **Title** | WEXTOR: a Web-based tool for generating and visualizing experimental designs and procedures | A framework for evaluating the data-hiding capacity of image sources | Chinese tourists step up for Abe as Japanese tighten belts | Manafort family business defends name as cousin sits in jail |
| **Abstract** | WEXTOR is a Javascript-based experiment generator and teaching tool on the World Wide Web that can be used to design laboratory and Web experiments in a guided step-by-step process. It dynamically creates the customized Web pages and Javascripts needed for the experimental procedure and provides experimenters with a print-ready visual display of their experimental design. WEXTOR flexibly supports complete and incomplete factorial designs with between-subjects, within-subjects, and quasi-experimental factors, as well as mixed designs. The software implements client-side response time measurement and contains a content wizard for creating interactive materials, as well as dependent measures (graphical scales, multiple-choice items, etc.), on the experiment pages... | An information-theoretic model for image watermarking and data hiding is presented in this paper. Previous theoretical results are used to characterize the fundamental capacity limits of image watermarking and data-hiding systems. Capacity is determined by the statistical model used for the host image, by the distortion constraints on the data hider and the attacker, and by the information available to the data hider, to the attacker, and to the decoder. We consider autoregressive, block-DCT, and wavelet statistical models for images and compute data-hiding capacity for compressed and uncompressed host-image sources. | When Jingyan Hou made her first trip to Japan in 1997, the office worker from Beijing spent ¥200,000 during a weeklong stay on accommodations, meals, transport and souvenirs. On her second visit this year, she spent that much on just one Louis Vuitton handbag in Tokyo’s Ginza shopping district. The increasing wealth of travelers like Hou, 45, underscores the opportunity for Japan to expand its tourism industry as China’s burgeoning middle class goes on vacations abroad. The yen’s slump to a seven-year low against the dollar is also broadening the country’s appeal globally and bolstering the Abe administration’s effort to double visitors by the 2020 Tokyo Olympics. “There’s a lot of room to boost the number of foreign tourists coming to Japan with these growing economies in our neighborhood,” said Daiki Takahashi, an economist at the Dai-ichi Life Research Institute in Tokyo... | What do you do if you share a name with one of the most prominent defendants in the special counsel’s investigation into Russia? Paul Manafort’s daughter decided to change her name. Leaders of New Britain, Connecticut, considered renaming Paul Manafort Drive, a street named after his father. At Manafort Brothers Inc., a family-owned New England construction firm, they are defending the Manafort name and legacy while distancing themselves from their cousin, Trump’s former campaign chairman who was recently blasted by prosecutors for years of lies and lawbreaking. The Manafort name has been a familiar one in New England politics and business for decades, creating a predicament for the family as the 69-year-old former attorney is scheduled to be sentenced Thursday. Manafort Brothers is one of New England’s best known construction companies... |
| **Ground Truth Keyphrases** | [’WEXTOR’, ’Web-based tool’, ’Javascript-based experiment generator’, ’teaching tool’, ’World Wide Web’, ’customized Web pages’, ’print-ready visual display’, ’factorial designs’, ’client-side response time measurement’, ’content wizard’, ’HTML’, ’Web server’, ’logfiles’, ’database’, ’experimental design visualization’, ’free software’] | [’data-hiding capacity’, ’image sources’, ’information-theoretic model’, ’watermarking’, ’capacity limits’, ’statistical model’, ’distortion constraints’, ’wavelet statistical models’, ’uncompressed host-image sources’, ’closed-form expressions’, ’sparse-model approximations’, ’geometric attacks’, ’distortion measures’, ’autoregressive statistical models’, ’block-DCT statistical models’, ’compressed host-image sources’ | [’tourism’, ’china’, ’tokyo 2020’, ’2020 olympics’, ’inbound tourism’] | [’paul manafort’, ’manafort brothers’, ’new england’, ’robert mueller’, ’donald trump’, ’russia probe’] |
| **Llama-3** | [’Web-based experiment generator’, ’experimental design’, ’factorial design’, ’response time measurement’, ’client-side’, ’HTML editor’, ’Web experimental design’, ’educational purposes’, ’noncommercial purposes’, ’psycholinguistics’] | [’Image watermarking’, ’data hiding’, ’information-theoretic model’, ’capacity limits’, ’image sources’, ’statistical models’, ’autoregressive’, ’block-DCT’, ’wavelet’, ’compressed’, ’uncompressed host-image’, ’sparse-model’, ’geometric attacks’, ’distortion measures’] | [’China’, ’Japan’, ’Abe’, ’tourism’, ’economy’, ’Tokyo Olympics’, ’travel’, ’shopping’, ’Louis Vuitton’, ’Ginza’, ’Beijing’, ’accommodation’, ’consumption tax’, ’recession’, ’World Tourism Organization’, ’South Korea’, ’Taiwan’, ’Tokyo’, ’Mount Fuji’, ’Kyoto’, ’Sakaiminato’, ’Tottori’, ’cruise ship’, ’visa’, ’Okinawa’, ’Indonesia’, ’Philippines’, ’Vietnam’, ’customer service’, ’transport’, ’cultural attraction’, ’hostel’, ’love hotel’, ’Asakusa’] | [’Paul Manafort’, ’Manafort Brothers’, ’construction company’, ’family business’, ’Russia investigation’, ’special counsel’, ’Russia’, ’politics’, ’business’, ’Connecticut’, ’New England’, ’Trump’, ’campaign chairman’, ’lawyer’, ’jail’, ’conviction’, ’sentencing’, ’fraud’, ’politics’, ’family name’, ’legacy’, ’reputation’, ’morals’, ’ethics’, ’philanthropy’] |
| **Phi-3** | ’Experimental design generator’, ’Web-based experiment tool’, ’Guided procedure design’, ’Dynamic content creation’, ’Flexible factorial design’, ’Client-side response measurement’, ’Content wizard’, ’Interactive materials design’, ’Methodological features’, ’Web server compatibility’, ’Educational resource’, ’Freely available’, ’World Wide Web’, ’GenPSYLab’, ’Login link’] | [’Information-theoretic model’, ’Image watermarking’, ’Fundamental capacity limits’, ’Statistical model’, ’Distortion constraints’, ’Data hider vs. attacker’, ’Compressed vs. uncompressed images’, ’Sparse model approximations’, ’Geometric attacks’, ’Distortion measures’, ’Invariant attacks’] | [’Chinese tourists growth’, ’Japanese tourism industry expansion’, "Abe administration’s doubling tourists", "yen’s slump", "Japanese economy’s recession", ’foreign visitors spending’, ’duty-free sales increase’, ’bilingual sales assistants’, ’tourism in less-visited areas’, "Tottori’s tourist infrastructure", ’visa policy changes’, ’inexpensive accommodations’, ’Manryo hostels’, ’foreign tourist increase’] | ’Manafort family business’, ’defend name’, ’Paul Manafort’, ’criminal issues’, ’Manafort Brothers Inc.’, ’strong moral and business ethics’, ’family legacy’, "Paul Manafort’s arrest", ’Paul Manafort Drive’, "Paul Sr.’s political scandals", "Paul Jr.’s fight to dissociate", ’minority subcontractor scandal’, ’Manafort family foundation’, "dissociation from name by Manafort’s daughter "] |
| **GPT-4o** | [’WEXTOR’, ’web-based tool’, ’experimental designs’, ’Javascript-based experiment generator’, ’laboratory experiments’, ’web experiments’, ’factorial designs’, ’between-subjects factors’, ’within-subjects factors’, ’quasi-experimental factors’, ’mixed designs’, ’client-side response time measurement’, ’content wizard’, ’interactive materials’, ’dependent measures’, ’graphical scales’, ’multiple-choice items’, ’web experimental design’, ’platform independent’, ’educational purposes’, ’noncommercial purposes’, ’visualization of experimental design’, ’experiment procedures’, ’customized web pages’] | [’data-hiding capacity’, ’image watermarking’, ’information-theoretic model’, ’image sources’, ’statistical model’, ’host image’, ’distortion constraints’, ’data hider’, ’attacker’, ’decoder’, ’autoregressive model’, ’block-DCT model’, ’wavelet model’, ’compressed host-image sources’, ’uncompressed host-image sources’, ’sparse-model approximations’, ’geometric attacks’, ’distortion measures’, ’capacity limits’, ’closed-form expressions’] | [’Chinese tourists’, ’Japan tourism industry’, "Shinzo Abe’s growth strategy", "yen’s slump", ’Tokyo Olympics 2020’, ’consumption tax increase’, ’foreign tourist spending’, ’inbound tourism’, ’consumer spending’, ’World Tourism Organization’, ’duty-free sales’, ’multiple-entry visas’, ’foreign tourists in Japan’, ’Manryo Inc. hostels’, ’Japanese economy’, ’cultural attractions’, ’Chinese middle class’, ’luxury shopping in Japan’, ’Sakaiminato tourism’, ’travel visas for Chinese tourists’] | [’Paul Manafort’, ’Manafort Brothers Inc.’, ’construction firm’, ’New England politics’, ’New Britain Connecticut’, ’special counsel’s investigation’, ’Trump’s former campaign chairman’, ’tax and bank fraud charges’, ’Ukrainian political interests’, ’Washington lobbying firm’, ’Black Manafort and Stone’, ’federal criminal and civil investigations’, ’minority subcontractor allegations’, ’perjury charges’, ’family legacy’, ’community foundation’, ’Jessica Manafort’, ’Jess Bond’, ’Rosy film’] |

Table 10: Two example input documents from Inspec (Reips and Neuhaus, 2002), (Moulin and Mihcak, 2002) and two from KPTimes (Times, 2014), (News, 2019), along with the corresponding keyphrase generations by various models, where blue represents present keyphrases and black represents absent keyphrases. These examples were chosen specifically to highlight the performance extremes across datasets: one demonstrating strong model performance and the other showcasing its limitations.