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

Anonymous ACL submission

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 in general, it is important to conduct a more thorough examination of their potential for keyphrase generation. In this paper, we attempt to meet this demand. Currently, we focus on the zero-shot capabilities of open-source instruction-tuned LLMs (Phi3, Llama3) for the 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. They serve as a high-level summary, providing quick insight into the text. Keyphrases can be "present keyphrases" if they appear verbatim in the text, or "absent", if they are semantically implied but do not occur explicitly in the text. While keyphrase extraction focuses on identifying present keyphrases (Bennani-Smires et al., 2018), keyphrase generation (KPG) extends the task to include both present and absent keyphrases (Meng et al., 2017; Yuan et al., 2020; Chan et al., 2019; Chen et al., 2020). Recent advancements in keyphrase research, including our 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).

A multitude of prior 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, 2024; 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, there 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 specialized to focus on specifically present keyphrases 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 get the best of both worlds. 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. Also, 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 early on. 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 others 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 (Ray 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 (in 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 multisampling aggregation strategies for KPG in §2.4 and demonstrate their corresponding results experimentally in §3.4.

We mainly focus on open-source instruction-tuned models, LLama 3 and Phi 3, in a zero-shot

Llama 3 & Phi 3

Llama3 Chat Template :
<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n{system_prompt}<|eot_id|><|start_header_id|>user<|end_header_id|>\n{user_prompt}\n**Title:** \"{title}\"\n\n**Abstract:** \"{abstract}\"\n\n **Instructions:** \n{instruction} <|eot_id|><|start_header_id|>assistant<|end_header_id|>\n[

Phi3 Chat Template :
<|system|>\n{system_prompt}<|end|>\n<|user|>\n{user_prompt}\n**Title:** \"{title}\"\n\n**Abstract:** \"{abstract}\"\n\n **Instructions:** \n{instruction} <|end|>\n<|assistant|>\n[

system_prompt:
You are an expert in finding the best keyphrases to represent the topical information of a scientific document just from reading its title and abstract.

user_prompt:
Generate present and absent keyphrases from the following title and abstract of a scientific document.

Instruction:
1. Only respond with a list of comma-separated keyphrases in the following format: \"[keyphrase 1, keyphrase 2, ..., keyphrase n]\"

Figure 1: Baseline Template used for Phi 3 and Llama 3

Order Control

Instructions:
1. List keyphrases in the order of their importance and relevance (from the most important and relevant to the least).
2. Only respond with a list of comma-separated keyphrases in the following format: "[keyphrase 1, keyphrase 2, ..., keyphrase n]"

Length Control

Instructions:
1. Generate only the most important and relevant keyphrases. Don't overgenerate..
2. Only respond with a list of comma-separated keyphrases in the following format: "[keyphrase 1, keyphrase 2, ..., keyphrase n]"

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.

context. 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 usefully 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) 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 and Phi 3. 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

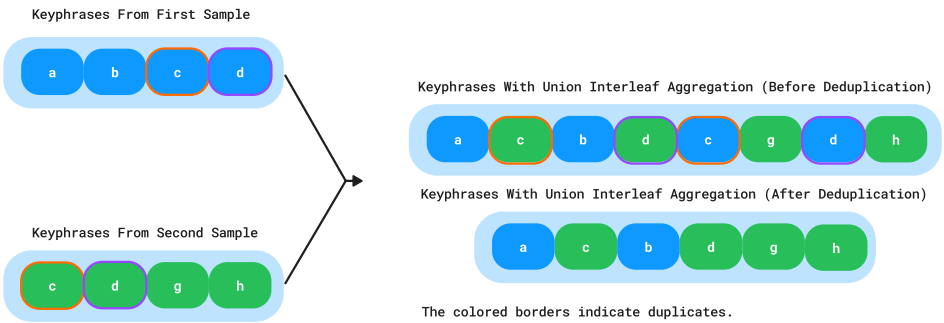

Figure 3: Visualization of Union Interleaf aggregation over multiple samples.

parenthesis "[" so that the models can directly start generating the keyphrases without any in-between fluff. 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. The first three are manually defined. We define them the same way for Llama 3 and Phi 3. Our definitions for `system_prompt`, the `user_prompt`, and the `instruction` variables are also shown in Figure 1. When evaluating the models on KP-Times, we change any occurrence of "scientific document" with "news article". The user prompt is roughly inspired from the TP4 prompt template in Song et al. (2023b). We use it because it presents a reasonable balance in their paper. In the baseline, the `instruction` merely provides some formatting specifications to make parsing keyphrase 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:

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).

Models	Inspec		Krapivin		SemEval		KP20k		KP-Times	
	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
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

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

2.4 Multisampling (RQ3)

To investigate RQ3, we stochastically generate multiple samples from LLMs using some temperature for diversity with the baseline prompt. We take independent samples like this (similar to self-consistency strategies (Wang et al., 2023)) because beam search (which is sometimes used in KPG) can be expensive with large LLMs and can also tend towards worse diversity.

In our multisampling 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. We sort the samplings 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 $F_1@5$. Thus we keep sorting 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 them and stemming them. These are standard normalization strategies also used for evaluation to determine which keyphrases are identical. We also deduplicate each sample while preserving the order.

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. We discuss them 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 (starting from the highest-ranked beam to the lowest) together. 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 com-

Models	Inspec		Krapivin		SemEval		KP20k		KP-Times	
	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
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
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

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

bined list in that order. The combined list will look like: $(k_1^1, k_1^2, \ldots, k_1^n, k_2^1, k_2^2, \ldots k_2^n, \ldots\ldots, k_m^n)$. 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 the 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 question as to how to dynamically select an appropriate number of keyphrases for the 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 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[1]. 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.

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

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

Models	Inspec		Krapivin		SemEval		KP20k		KP-Times	
	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
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
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**

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

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 aggregated list.

3 Experiments and Results

For our experiments, we choose a temperature of 0.8 which we use consistently[2] across all models and datasets. We explain our evaluation in the 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.

3.2 Specialist Prompts Results (RQ1)

In Table 1, we present the results of our baseline and specialist (present and absent) prompts.

[2]We chose 0.8 because it is in the standard range of temperature typically used for self-consistency for diverse multi-sampling. We also did not find substantial differences from different temperatures in a subset of the KP20K validation set.

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. Thus, at least for the explored LLM-based models and the considered prompts, the answer to RQ1 seems to be negative. Given that the individual specialists do not show promising results at the moment over the simple baseline, we did not further explore ensembling strategies to combine the specialists.

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 Multisampling Results (RQ3)

In Table 3, we present the results of multisampling with various aggregation strategies. As we would expect, simple union do not help much, and often may harm the performance because they remove 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 multisampling aggregation. Among the three, Frequence Order-based aggregation most consistently shows the best performance; particularly, on absent keyphrase generation. Overall, we find that the best aggregation methods with multisampling significantly improve the performance of LLMs over the baseline. As such, the answer to RQ3 leans towards an affirmation. In the Appendix Table 5, we also show how well absent keyphrases are recalled for the multisampling-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, even if not state of the art, 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.

Note that our intention in this paper was to do a focused study on potential unexplored ways to improve zero-shot LLM performance for KPG, rather than integrating all the best-known strategies to get the SOTA results.

4 Related Work

Identifying keyphrase from a document is a long-standing task and has been well studied in the literature using both supervised and unsupervised approaches. However, with the surge of deep learning models, the attention has shifted towards generative models particularly because of their capability to generate absent keyphrases (Meng et al., 2017). Many recent works for keyphrase generation have explored seq2seq model 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; Ray Chowdhury et al., 2022; Madaan et al., 2022; Wu et al., 2023, 2024). 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). In our paper, we explore LLMs using novel strategies such as "specialist prompts", task-specific instructions, and

Models	Inspec		Krapivin		SemEval		KP20k		KP-Times	
	F1@M	F1@5	F1@M	F1@5	F1@M	F1@5	F1@M	F1@5	F1@M	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 (Ray 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 Multisample	49.9	45.6	31.8	33.6	38.0	38.1	28.7	32.1	24.7	31.5
Phi 3 Multisample	**54.7**	**50.9**	25.1	24.7	32.9	30.5	19.7	20.4	12.0	11.9
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 (Ray 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 Multisample	8.5	7.6	5.9	5.1	3.9	3.6	5.4	4.9	5.3	5.0
Phi 3 Multisample	**9.3**	**9.0**	2.1	2.1	2.5	2.7	2.0	2.0	0.6	0.7

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 Multisample denotes Llama3/Phi3 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 ± 0.1.

multi-sampling and contrast them with many of these works.

5 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 multisampling 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 multisampling 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 multisampling aggregation strategies are also model-agnostic and could be explored with earlier established KPG models as well as with beam search. We leave this for future work.

6 Limitations

Some of the findings (e.g., related to RQ1 and RQ2) may not apply to newer and/or larger LLMs. Moreover, while multi-sampling boosts the performance, it will also come with added latency and computation. Despite these limitations, we believe our LLM-based methods show promise and offer a strong foundation for future work in LLM-based keyphrase generation.

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.

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.

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.

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*.

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. 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.

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.

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

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*.

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.

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. 2024. 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.

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.

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 metrics calculate the F_1 between all the predicted keyphrases by the model and the ground truth keyphrases. In case of multisampling models "all the predicted keyphrases" stands for all the keyphrases remain after selection 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 metrics calculate 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.

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

4. R@Inf: This evaluation metrics calculate the recall between the 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 multisample models, for @Inf, we don't 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 lowercased stemmed (using Porter-Stemmer) version of keyphrases match with the lowercased stemmed version of the input text.

Models	Inspec		Krapivin		SemEval		KP20k		KP-Times	
	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.