Improving Absent Keyphrase Generation with Diversity Heads

Edwin Thomas*
University of Ottawa
Canada
ethom123@uottawa.ca

Sowmya Vajjala
National Research Council
Canada
sowmya.vajjala@nrc-cnrc.gc.ca

Abstract

Keyphrase Generation (KPG) is the task of automatically generating appropriate keyphrases for a given text, with a wide range of real-world applications such as document indexing and tagging, information retrieval, and text summarization. NLP research makes a distinction between present and absent keyphrases based on whether a keyphrase is directly present as a sequence of words in the document during evaluation. However, present and absent keyphrases are treated together in a text-to-text generation framework during training. We treat present keyphrase extraction as a sequence labeling problem and propose a new absent keyphrase generation model that uses a modified cross-attention layer with additional heads to capture diverse views for the same context encoding in this paper. Our experiments show improvements over the state-of-the-art for four datasets for present keyphrase extraction and five datasets for absent keyphrase generation among the six English datasets we explored, covering long and short documents.

1 Introduction

Predicting keyphrases in a given text is useful in many application scenarios related to indexing and information retrieval. While some of the research on this topic focused on extracting only the keyphrases that directly appear in the document i.e., present keyphrases (See Song et al. (2023a) for a survey of extractive methods), more recent work on this topic treated it as an end-to-end text generation task, covering both present and absent keyphrases i.e., those that are not directly seen in the text. Existing approaches can be categorized into three text representation paradigms: *one2one*, where the model learns to generate one keyphrase at a time per text (Meng et al., 2017); *one2seq*, where all the keyphrases are generated as a single delimiter-separated sequence (Yuan et al., 2020); and one2set, where the model generates the keyphrases as a set (Ye et al., 2021b).

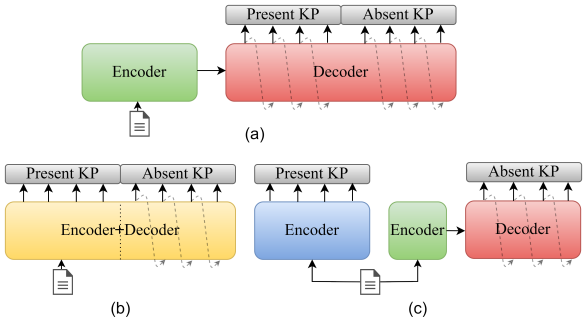

Figure 1: Different Types of Architecuture Configurations for the KPE task. Configurations (a), (b) and (c) represents TransSet, UniKP and the proposed work respectively.

Architecturally, KPG research falls into three categories, depicted in Figure 1. A common approach is to treat both present and absent keyphrases together, generating them autoregressively (e.g., Yuan et al., 2020), and Figure 1 (a) illustrates this. A small number of approaches treat extraction and abstraction separately, but in a multi-task learning setup (e.g., Wu et al., 2021; Ahmad et al., 2021) where extraction is treated as sequence labeling and abstraction as autoregressive generation, illustrated by Figure 1 (b). Treating them as completely distinct (Figure 1 (c)) has the potential to achieve increased performance on both, which we explore in this paper.

A recurring problem in KPG research is that of repetition and duplication among the generated keyphrases. Various approaches have been proposed in the literature to address this issue, which often result in high recall, low precision scenarios. An absent keyphrase generation approach that offers a good trade-off between precision and recall while generating diverse keyphrases is needed,

* Work done during a summer internship at National Research Council, Canada

Findings of the Association for Computational Linguistics: NAACL 2024, pages 1568–1584
June 16-21, 2024

which is also addressed in this paper.

To summarize, we treat present keyphrase extraction and absent keyphrase generation as separate tasks, and focus on improving the diversity of absent keyphrase generation by introducing what we call *diversity heads* in this paper. The main contributions of the paper are listed below:

1. We propose a new *one2set* model for absent keyphrase generation that uses a modified cross-attention layer that leverages additional heads to capture diverse views for the same context encoding, and achieve state-of-the-art absent keyphrase generation performance on 5 of the 6 test sets.

2. By separating present keyphrase extraction and absent keyphrase generation into independent tasks, we show better performance on both tasks for 4 of the 6 test sets we tested with.

The rest of the paper provides some background (Section 2), describes our algorithm (Section 3) and our experimental setup (Section 4), followed by a discussion of our results (Section 5), main conclusions (Section 6) and brief notes on the limitations, and an ethics statement (Section 7 and 8).

2 Related Work

KPG was first introduced as a task by Liu et al. (2011), who applied statistical machine translation techniques to keyphrase extraction to be able to generate phrases that did not directly appear in the document. Since then, it has been predominantly treated as a supervised learning problem in the literature, although some research on unsupervised approaches exists (Shen et al., 2022). We will focus on supervised approaches in this paper.

Meng et al. (2017)'s CopyRNN is perhaps the first to adapt a sequence-to-sequence generation approach to KPG, which resulted in a new wave of neural keyphrase generation models (Chen et al., 2018; Zhang and Xiao, 2018; Chen et al., 2019b; Yuan et al., 2020; Chen et al., 2020, etc.). Generative adversarial networks (Swaminathan et al., 2020; Lancioni et al., 2020), variational autoencoders (Santosh et al., 2021a), graph neural networks (Ye et al., 2021a), multi-task learning (Ye and Wang, 2018; Chen et al., 2019a; Koto et al., 2022; Zahera et al., 2022), reinforcement learning (Chan et al., 2019; Luo et al., 2021), contrastive learning (Choi et al., 2023) and pre-trained language models have all been explored for KPG (Wu et al., 2021, 2022a; Chowdhury et al., 2022; Diya and Mizuho, 2022; Kundu et al., 2023; Wu et al., 2023b, 2024).

A recent strand of research modeled KPG as a set generation problem, instead of sequence generation, to suit the nature of KPG better instead of forcing it into the seq-to-seq setting (Ye et al., 2021b; Xie et al., 2022). Improving sequence to sequence architectures with additional information using topic models (Wang et al., 2019; Zhou et al., 2021), gazetteers (Santosh et al., 2021b), incorporating linguistic constraints (Zhao and Zhang, 2019), addressing the semantic bias in training (Zhao et al., 2022), developing structure aware representations that consider additional textual information such as title or full-text along with abstract (Chen et al., 2019b; Wang et al., 2020; Kim et al., 2021) were all explored in past research.

Some research explicitly separate present and absent keyphrase modeling with a multi-task learning framework that jointly learns to extract (present) as well as and generate (absent) keyphrases (Chen et al., 2019a; Ahmad et al., 2021; Wu et al., 2021), and a few approaches just treat them as separate tasks (Liu et al., 2021; Wu et al., 2022b). Other recent research reported comparisons of contemporary KPG approaches with zero-shot performance of ChatGPT (Martínez-Cruz et al., 2023; Song et al., 2023b). Xie et al. (2023) presents a comprehensive survey of existing research on this topic.

One major issue KPG is repetition and duplication. Beam search decoding is commonly used to over-generate and consider the top-ranked keyphrases, along with additional strategies. Yuan et al. (2020) described a semantic coverage mechanism to increase the diversity by altering the decoder hidden states. Chen et al. (2020) proposed a hierarchical decoding mechanism which keeps track of previously generated keyphrases and increases diversity. Bahuleyan and El Asri (2020) addressed this issue by training the keyphrase generation model with a neural unlikelihood objective and Ahmad et al. (2021) employed sentence selection and layer-wise coverage attention to increase diversity among the generated keyphrases, along with beam search. Ye et al. (2021b); Xie et al. (2022) address the diversity issue by treating the problem as sequence-to-set generation instead of sequence-to-sequence. In this paper, we propose a

new architecture for absent keyphrase generation and address this issue using what we call as *diversity heads*.

3 Our Approach

We separate present keyphrase extraction and absent keyphrase generation in this paper, by treating the former as sequence labeling, and the latter as sequence-to-set generation. Details of our approach are explained below.

3.1 Present Keyphrase Extraction

We define Present Keyphrase (PKP) extraction as sequence labeling using the standard BIO encoding scheme, and use an additional label X to denote subword tokens within a keyphrase, following Wu et al. (2021). Additionally, we address the inherent class imbalance of sequence labels (as most tokens have a O label, indicating "other") by using a weighted negative log-likelihood loss function based on mini-batch statistics (Eq. (1)) where the weight per token label is defined as the inverse of its frequency in the mini-batch (Eq. (2)).

$$\mathcal{L}^{SeqLab} = -\sum_{s=1}^{S} w_s^{(b)} log\left(\frac{e^{y_s^{(b)}}}{\sum_{c=1}^{C} e^{y_{s,c}^{(b)}}}\right) \quad (1)$$

$$w_s^{(b)} = \left(\frac{1}{|B| * S}\sum_{n=1}^{|B|*S} \mathbb{1} \cdot [\hat{y}_n = \hat{y}_s^{(b)}]\right)^{-1} \quad (2)$$

Here, S is the maximum number of tokens in any given instance b for a batch B and C is the number of sequence labels defined using the BIO-X scheme i.e., 4.

Note that while some previous approaches do treat PKP as sequence labeling (Chen et al., 2019a; Liu et al., 2021; Wu et al., 2021; Ahmad et al., 2021; Wu et al., 2022b), they still treat the extraction and generation tasks either in a joint-learning setup or as a cascade where the sequence labeling output informs the generation step. We just treat the two as independent tasks.

3.2 Absent Keyphrase Generation

We propose a new sequence-to-set generation model that improves the diversity of the generated Absent Keyphrases (AKP) using specialized attention heads. The absent keyphrase generation task is defined formally as follows: Let D represent the sequence of input document tokens and $\hat{Y} = \{y_1, y_2, ..., y_n\}$ the target set of keyphrases to be generated. If K is the maximum number of keyphrases generated by the transformer decoder, and S the maximum keyphrase length per unit, then the goal is to optimize the Encoder weights (θ_e) and Decoder weights (θ_d) to generate a prediction set $Y = \{y_1, y_2, \ldots, y_k\}$, that is closest to the target keyphrase set $\hat{Y}$ where $|y_k| \leq S$ and $k \leq K$. Our approach (depicted in Figure 2) adapts the transformer model, which is only capable of sequence-to-sequence generation, to predict a set instead by using a permutation-invariant training strategy.

Permutation Invariance To enforce target order invariance during training, we employ the Hungarian algorithm (Kuhn, 1955) to first re-align the targets to the corresponding decoder keyphrase unit predictions following Ye et al. (2021b). The optimal permutation $\hat{\sigma}(k)$ at any given training step is given by Eq. (3), where $m = min(|\hat{y}^{\sigma(k)}|, S)$ and y_b^a denotes the b^{th} token of the a^{th} KP unit and $y_{b,c}^a$ represents the probability that this token corresponds to the c^{th} token in the vocabulary.

$$\hat{\sigma}(k) = \underset{\sigma \in S(K)}{\text{argmax}} \sum_{k=1}^{K}\sum_{s=1}^{m} \mathbb{1} \cdot [\hat{y}_s^{\sigma(k)} \notin S_\phi] y^k_{s,\hat{y}_s^{\sigma(k)}} \quad (3)$$

The keyphrase sets at any given training step can be modeled as a complete weighted bipartite graph where the nodes at each side correspond to the target tokens $\hat{y}$ and predicted tokens y, and the edge weights correspond to the likelihood of predicted keyphrase matching the target. The edge weights are computed by summing over the predicted keyphrase token-level likelihoods where the tokens for extracting the likelihood scores is obtained from the $\sigma(k)^{th}$ target keyphrase permutation index. The encoder-decoder model can then be optimized by minimizing the negative log-likelihood of the softmax scores as shown in Eq. (4) where K_v is the set of all indices of non-empty keyphrase units and V is the vocabulary.

$$\mathcal{L}^{EncDec} = -\sum_{k \in K_v}\sum_{s=1}^{m} log\left(\frac{e^{y^k_{s,\hat{y}_s^{\hat{\sigma}(k)}}}}{\sum_{c=1}^{|V|} e^{y^k_{s,c}}}\right) \quad (4)$$

Diversity Heads: In the traditional encoder-decoder architecture, the input key and value projections from the encoder are consistent across all the generated keyphrases which limits the diversity. This effect is amplified in the set generation setting where independent decoding restricts the

model from learning the relationships with previously generated keyphrases and subsequently reduces repetitions. To increase the diversity of the generated keyphrases, we propose a modified cross-attention layer that leverages additional heads to capture diverse views for the same context encoding. We introduce separate key and value attention modules for each decoder keyphrase unit to achieve this. As seen in Eq. (5), $W_{ik}^{(l)}$ is used to learn different encoder representations for the l^{th} layer, and for each of the k units respectively. The attention function Vaswani et al. (2017), is then computed over the packed diversity head keys and values, and the queries as shown in Eq. (6).

$$\begin{aligned} DivK_i^{(l)} &= Concat(K_1^{(l)}W_{i1}^{(l)K}, ..., K_k^{(l)}W_{ik}^{(l)K}) \\ DivV_i^{(l)} &= Concat(V_1^{(l)}W_{i1}^{(l)V}, ..., K_k^{(l)}W_{ik}^{(l)V}) \\ Q_i^{(l)} &= Concat(Q_1^{(l)}W_i^{(l)Q}, ..., Q_k^{(l)}W_i^{(l)Q}) \end{aligned} \tag{5}$$

$$H_i^{(l)} = Softmax\left(\frac{Q_i^{(l)} * {DivK_i^{(l)}}^T}{\sqrt{d_k}}\right) * DivV_i^{(l)} \tag{6}$$

The resultant projections are then concatenated to collectively focus on different representation subspaces obtained by the h attention heads (Eq. (7)).

$$DivCrossAttn = Concat(H_i^{(l)}, ..., H_h^{(l)})W_F \tag{7}$$

The self-attention mechanism follows Vaswani et al. (2017) for both the encoder and the decoder. Figure 2 gives a high level depiction of the proposed architecture.

3.2.1 Training Methodology

The training process can be divided into three major steps A.1, A.2 and A.3 as described in Algorithm 1. We start with the encoder input sequence $x_{1:n}$, max AKP units K, max keyphrase length S, decoder target $y_{1:k*s}$, decoder positional ids $pid_{1:k*s}$, iterations T and batch size B. At each step, the contextual embeddings h^e is obtained using a single forward pass through the encoder (A.1). The realignment of the target is performed by ignoring the gradient computations in the first decoder forward pass block (lines 4-10). Student forcing is used to obtain the hidden states l^{d_s} (A.2), and subsequently the best permutation of the target keyphrases σ (Eq. (3)), is used to realign the target (line 7). This is followed by the second stage (A.2), where the reordered target and encoder hidden states are used

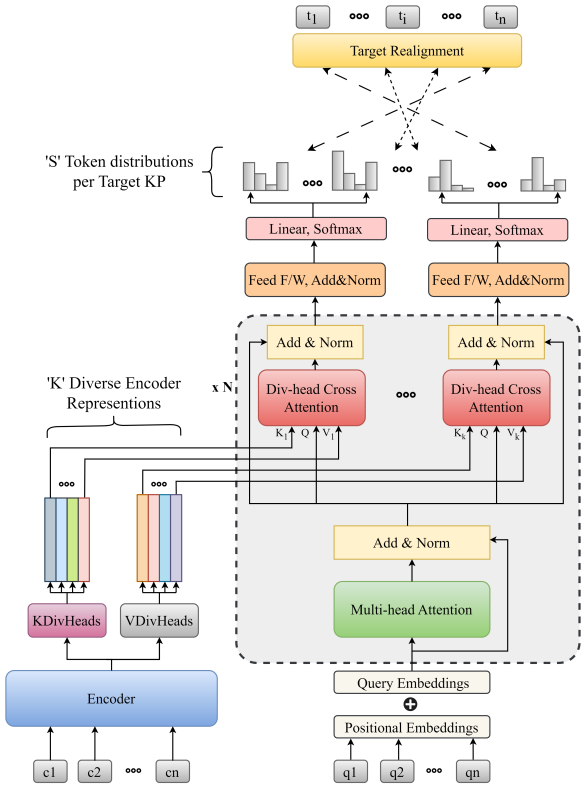

Figure 2: Proposed Architecture with Diversity Heads and Target Realignment.

to train the AKP model using teacher forcing and the encoder-decoder Loss as described in Eq. (4)).

Algorithm 1: AKP Model Training Cycle

1 **repeat** T **times**
2 **for** $b = 1, \ldots, B$ **do**
3 $h^e_{1:k*s,1:hdim}$ = Encoder($x^{(b)}_{1:n}$)
4 **with** $gradientTracking = False$ **do**
5 $h^{d_s}, l^{d_s}_{1:k*s,1:|V|}$ = DivHeadsDecoder($h^e, pid_{1:k*s}$) ; ▷ (A.1)
6 $\sigma_{1:k}$ = LinearAssignment($l^{d_s}_{1:k*s,1:|V|}, y_{1:k*s}$)
7 $\widetilde{y}_{1:k*s}$ = Realign($\sigma_{1:k}, y_{1:k*s}$)
8 **end**
9 $h^{d_t}, l^{d_t}_{1:k*s,1:|V|}$ = DivHeadsDecoder($h^e, \widetilde{y}_{1:k*s}$) ; ▷ (A.2)
10 $\mathcal{L}$= $\mathcal{L}^{EncDec}(\widetilde{y}_{1:k*s}, l^{d_t})$
11 **end**
12 **end**

It is to be noted that the additional latency due to the two-stage training process does not translate to the inference stage as the first forward pass for realigning the targets is exclusive to the training cycle. The Greedy search decoding strategy coupled with the diversity heads can result in improved diversity of generated absent keyphrases, without using costly search techniques such as beam search dur-

ing inference. Additionally, we also note that while the original Seq2Set approach (Ye et al., 2021b) has a higher inference latency due to costly auto-regressive decoding of both the AKP and PKP sequences, our approach optimizes the inference latency by treating extraction as a separate, sequence labeling task[1].

4 Experimental Setup

This section describes the implementation details, and provides the details of the datasets used, evaluation measures, and the baselines we compare against.

Implementation Details All experiments were conducted in a distributed training setup with four NVidia Tesla V100 GPUs, and run for 5 epochs, with a batch size of 128 (per-gpu batch size of 32), and a text length of 384 tokens. The decoder is configured to have 8 independent decoder units with a maximum keyphrase length of 5 tokens (total 40 output tokens). These numbers were set after an empirical analysis with the validation datasets. The model is based on the transformer encoder-decoder backbone (Vaswani et al., 2017) with 12 layers. A learning rate of 3e-05 is used with the Adam Optimizer (Kingma and Ba, 2014) and a non-linear learning rate scheduler with warm-up proportion of 0.1.

Datasets We used two datasets for training, and six datasets (including the test splits of training datasets) for testing in our experiments. All the datasets are in English, and we used the partitions used in previous research. They are described below:

1. KP20K (Meng et al., 2017) is a large dataset of ∼530k scientific paper abstracts from the computer science domain, with author assigned keyphrases, and is the most commonly used dataset to train KPG models. We used the official train/validation splits for training and test split for evaluation.

2. KPTIMES consists of ∼290k news articles with expert labeled keyphrases (Gallina et al., 2019), and we used the official train/validation splits for training and test split for evaluation.

3. INSPEC (Hulth, 2003) is a test set with 500 computer science abstracts annotated by professional indexers.

4. SEMEVAL (Kim et al., 2010)'s test set consists of 100 full length articles from ACM digital library with student and expert annotated keyphrases.

5. KRAPIVIN (Krapivin et al., 2009) consists full text computer science articles with author annotated keyphrases.

6. NUS (Nguyen and Kan, 2007) consists of full text scientific articles with author assigned, and externally annotated keyphrases by student volunteers.

Table 1 shows some basic statistics about the datasets. For models trained on KP20K, we tested using KP20K's test split, and the four test sets - Inspec, Krapivin, Semeval and NUS, as is commonly done in practice. For models trained on KPTimes, we evaluated only on the test-split of KPTimes, as they are all from a different domain, with less than 5% overlap between the labeled keyphrases in KPTimes and other test sets[2].

Dataset	#docs	#pkp/doc	#akp/doc
Train			
KP20K-Train	∼530k	2.34	2.94
KPTimes-Train	∼ 260k	2.15	2.88
Test			
KP20K-Test	20000	2.34	2.93
KPTimes-Test	20000	2.72	2.31
Inspec	500	6.57	3.26
SemEval	100	9.2	6
Krapivin	2304	3.73	1.6
NUS	211	8	3.07

Table 1: Dataset statistics

Evaluation: Evaluation measures in KPG are not uniform and a range of measures (recall, f-score, MAP, etc.) are reported. In this paper, we report a commonly used measure, macro-F1@M, where M is the number of generated keyphrases, and a less commonly used macro-F1@O, where O is the

[1]Our implementation code is provided as supplementary material at: https://github.com/edwinthomas444/diverse-keyphrase-generation

[2]More detailed statistics are provided in the appendix Table A1 and overlap statistics among the datasets are provided in Table A2

number of ground truth keyphrases and compare them later in Section 5.5 [3].

Baselines: We re-trained three recent keyphrase generation models that shared code repositories publicly - UniKeyphrase (Wu et al., 2021), KPDrop (Ray Chowdhury et al., 2022) and One2Set (Ye et al., 2021b), and these are used as external baselines for both present and absent keyphrases. KPDrop is a model-agnostic technique, and we use the KPDrop-A with One2Set model, which achieved the best results in their experiments. Retraining of all the three approaches was done for 5 epochs, keeping the rest of the settings unchanged, to keep the number of epochs consistent across all experiments. Additionally, we also compare with other recent results (without re-implementation, citing as is). For absent keyphrases, we also do ablation tests comparing our diversity heads approach to sequence-to-sequence generation and sequence-to-set generation without diversity heads, and also discuss the effect of increased model parameters in our approach. Additionally, while greedy decoding is our default, we also report results with beam search for comparison. Note that our absent keyphrase generation model is trained only on that part of the data, unlike the other approaches mentioned above, which treat present and absent keyphrases together for generation.

5 Results and Discussion

We present the main findings about present keyphrase extraction and absent keyphrase generation, followed by an analysis of the results in this section. Considering the multiple train/test sets, we separate the discussion by short (abstracts) vs long (full text) documents. The first three approaches we compare against (UniKP, TransSet and KPDrop-A in Tables 2– 5) were replicated and re-run and the next three (zero-shot ChatGPT, SEG-Net and Wu et al. (2023a) are taken as-is from the cited papers.

5.1 Present Keyphrase Extraction

Tables 2 and 3 summarize the performance of our present keyphrase extraction approach in comparison with other existing models respectively. Treating extraction as separate from abstraction (and hence, generation) seems to show clear benefits for longer documents with our model achieving the best results on all the four datasets in terms of F1@M and on three of them in terms of F1@O (Table 3). There is an over 5% improvement over the state-of-the-art for one of the datasets (SEMEVAL). For shorter texts, while our approach performed comparably to the best approach on KP20K, the performance on INSPEC, was poor, especially in terms of F1@M. We compare the evaluation measures in Section 5.5.

5.2 Absent Keyphrase Generation

Tables 4 and 5 present a comparison of our approach with other recent models in terms of absent keyphrase generation. Our model outperforms the state-of-the-art for both short and long texts on five of the six datasets we used, in terms of F1@O and four of the six datasets in terms of F1@M. There is a stark difference between the performance of models trained on KP20K and KPTimes, though. While the KPTimes model has a 10-15% drop between present and absent keyphrase performance, the KP20K trained model >30% drop between both cases, when tested on its own test partition. This was also reflected in the results for other test datasets the model was tested on. For example, on NUS, the present keyphrase performance was 48.59 and 43.09 in terms of F1@M and O respectively, whereas the absent keyphrase performance was 7.72 and 6.82 respectively, and in both cases, our model gave the best results compared to others. We discuss this issue briefly in Section 5.5.

5.3 Effect of Diversity Heads

While the results on absent keyphrase generation establish the merits of our method, they do not tell much on how better is the model when compared to plain sequence-to-sequence generation or sequence-to-set generation without diversity heads. Table 6 shows the results of this comparison for the two test datasets we used.

Since the differences (especially for KP20K) are under 1% in some cases, we conducted tests for statistical significance using bootstrap and permutation tests[4] following the guidance of Dror et al. (2018) on applying appropriate tests for NLP tasks. our proposed approach (seq2set+diversity heads) was significantly better than plain sequence-to-sequence generation as well as sequence-to-set generation for models trained on both the datasets, for both the evaluation measures ($p < 0.001$), with both the tests.

[3] Additional evaluation measures (P/R/F@5,M,O) are provided in the supplementary material in Tables A3– A7.

[4] https://github.com/rtmdrr/testSignificanceNLP

Trained on: KP20K-Train				
Model	KP20K-Test		Inspec	
	F1@M	F1@O	F1@M	F1@O
UniKP (Wu et al., 2021)*	27.19	38.21	13.85	14.35
TransSet (Ye et al., 2021b)*	37.16	29.53	31.97	**30.16**
KPDrop-A (Ray Chowdhury et al., 2022)*	38.42	30.85	30.06	29.18
zero-shot ChatGPT (Martínez-Cruz et al., 2023)	25.1	-	**40.3**	-
SEG-net (Ahmad et al., 2021)	37.9	-	26.5	-
Wu et al. (2023a)	**43.1**	-	40.2	-
Our Model	41.65	**41.81**	32.14	28.6

Table 2: Present Keyphrase Extraction for short documents (* indicates our reproduced results.)

Training Data	KP20k-Train						KPTimes-Train	
Test Data	Krapivin		SemEval		NUS		KPTimes-Test	
Model	F1@M	F1@O	F1@M	F1@O	F1@M	F1@O	F1@M	F1@O
UniKP (Wu et al., 2021)*	25.8	34.23	30.68	39.8	39.67	**47.17**	34.49	53.29
TransSet (Ye et al., 2021b)*	36.19	27.43	34.31	33.31	42.2	38.11	54.77	49.34
KPDrop-A (Ray Chowdhury et al., 2022)*	35.26	28.74	31.01	30.24	42.4	38.58	55.49	49.91
zero-shot ChatGPT (Martínez-Cruz et al., 2023)	-	-	18.6	-	19.96	-	29.0	-
SEG-net (Ahmad et al., 2021)	36.6	-	33.2	-	46.1	-	48.1	-
Wu et al. (2023a)	35.2	-	34.1	-	44.9	-	-	-
Our Model	**37.35**	**39.33**	**39.59**	**40.5**	**48.59**	43.09	**56.34**	**56.47**

Table 3: Present Keyphrase Extraction for long texts (* indicates our reproduced results)

Trained on: KP20K-Train				
Model	KP20K-Test		Inspec	
	F1@M	F1@O	F1@M	F1@O
UniKP (Wu et al., 2021)*	1.87	1.87	1.6	1.6
TransSet (Ye et al., 2021b)*	4.22	3.64	2.01	1.59
KPDrop-A (Ray Chowdhury et al., 2022)*	5.57	4.71	2.12	**2.12**
zero-shot ChatGPT (Martínez-Cruz et al., 2023)	4.4	-	**4.9**	-
SEG-net (Ahmad et al., 2021)	3.6	-	1.5	-
Wu et al. (2023a)	7.6	-	3.6	-
Our Model	**7.84**	**6.9**	1.18	0.89
Our Model (Beam search, n=5)	**9.68**	**9.4**	2.11	2.02

Table 4: Absent Keyphrase Generation for short documents (* indicates our reproduced results.)

Training Data	KP20k-Train						KPTimes-Train	
Test Data	Krapivin		SemEval		NUS		KPTimes-Test	
Model	F1@M	F1@O	F1@M	F1@O	F1@M	F1@O	F1@M	F1@O
UniKP (Wu et al., 2021)*	2.88	2.88	0.43	0.43	1.83	1.83	20.78	20.49
TransSet (Ye et al., 2021b)*	5.15	4.88	2.85	2.85	4.10	4.17	41.02	35.55
KPDrop-A (Ray Chowdhury et al., 2022)*	6.96	6.51	4.06	4.06	5.54	4.51	42.64	37.68
zero-shot ChatGPT (Martínez-Cruz et al., 2023)	-		2.1	-	4.2	-	2.2	-
SEG-net (Ahmad et al., 2021)	3.6	-	3.0	-	3.6	-	23.7	-
Wu et al. (2023a)	**8.6**	-	4.0	-	6.8	-	-	-
Our Model	7.59	**7.42**	**4.21**	**4.21**	**7.72**	**6.82**	**44.12**	**41.18**
Our Model (Beam search, n=5)	8.15	**7.98**	**4.79**	**4.79**	**8.18**	**7.02**	**43.26**	**44.13**

Table 5: Absent Keyphrase Generation results for long texts. (* indicates our reproduced results).

In KPG research, beam search is preferred over greedy search during inference, to improve recall and the diversity of the generated keyphrases (Meng et al., 2021). Table 7 shows how using diver-

	KP20K		KPTimes	
	F1@M	F1@O	F1@M	F1@O
Seq2Seq	6.25	6.14	41.24	39.71
Seq2Set	6.31	5.81	36.98	35.5
Our model	**7.84***	**6.9***	**44.12***	**41.18***

Table 6: Effect of Diversity Heads on AKP performance (* indicates that the result is significantly better than both the other approaches)

sity heads achieves better recall than a plain seq2set approach's beam search decoding even with greedy search. Using beam search with our approach results in slight improvements in recall compared to greedy search, although with longer inference time.

	Seq2Set		Ours	
	Greedy	Beam (n=5)	Greedy	Beam (n=5)
	R@K	R@K	R@K	R@K
KPTimes	29.64	30.22	**45.22**	**47.68**
KP20K	5.89	7.38	**9.11**	**11.71**
Krapivin	6.78	6.42	**8.25**	**10.32**
Inspec	1.04	1.72	**1.16**	**2.7**
SemEval	2.75	2.07	**2.74**	**3.25**
NUS	3.98	4.22	**7.48**	**7.65**

Table 7: Comparison of Recall@K (K=5) for Greedy versus Beam decoding

The comparisons in Tables 6 and 7 clearly show that the insertion of diversity heads results in a better performance across datasets, both in terms of overall F1 score as well as recall, even with greedy decoding. However, an alternative explanation for these better results could just be that our approach has much more parameters due to using different head weights for different decoder units in cross-attention, which would naturally(?) lead to more representational power and better performance. To explore this further, we performed some parameter scaling experiments, described in the next subsection.

5.4 Scaling the Model Parameters

We scaled the two seq2set models without diversity heads - plain Seq2Set model (second row in Table 6) and the TransSet model (second row in Table 4 to approximately match that of our proposed model with diversity heads[5]. The scaling operation

[5]We did not do this with the other Seq2Set model we replicated, namely KPDrop-A, because it uses TransSet as the base model.

model	F1@M	F1@O
Scaled Seq2Set	6.51(6.31)	6.47(5.81)
Scaled TransSet	4.6 (4.22)	4.1(3.64)
Our model (greedy)	7.84	6.9
Our model (beam, n=5)	9.68	9.4

Table 8: Comparison of scaled models. The unscaled performance is shown in parantheses

was achieved by increasing the number of heads in the cross-attention layer of the two seq2set models. To ensure that the representational power of the network is not changed after the scaling operation, a linear transformation layer is introduced after the cross-attention mechanism in the baseline seq2set model to reduce the embedding dimensions to size 768 (that matches our architecture). Table 8 shows the results on KP20K-TEST dataset (all scaled models are trained on KP20K-TRAIN).

We can observe that the scaled versions of Seq2Set and TransSet achieve slightly better results than their original versions, but are still not closer to our model. That leads us to a conclusion that the performance gain is not "just" because the model has more parameters, but is potentially because of the placement of diversity heads within the model architecture. [6]

5.5 Analysis of Results

In this section, we discuss two specific aspects of the results in more detail: the difference between the two performance measures used, and the performance difference between the two training datasets used.

F1@M versus F1@O We compared different methods using two evaluation measures F1@M and F1@O, where M and O refer to the number of predicted and ground-truth keyphrases respectively. Since we don't know the number of predicted phrases before hand, we could expect that F1@O will give lower numbers than F1@M, and we indeed notice it in many cases across Tables 2-6, but there are several interesting observations.

In present keyphrases, (Tables 2 and 3), UniKP is the exceptional case where F1@O consistently reports much higher numbers in present keyphrase extraction (e.g., almost 20% for KPTIMES-TEST)

[6]Detailed results with other test sets for both these models are in Appendices A8 and A9 for scaled Seq2Set and scaled TransSet respectively.

and TransSet and KPDrop report a drop for F1@O (almost 10% in some cases). One explanation could be that the number of keyphrases UniKP model outputs is closer to the number of ground truth keyphrases. However, it appears from the original paper (Wu et al., 2021, see Table 3) that UniKP does generate more keyphrases than other approaches compared there. Hence, an alternative explanation could be that UniKP generates better keyphrases in the top-K, but over-generates several other keyphrases, resulting in a better F1@O but worse F1@M. We leave more detailed explorations into the phenomenon for future work.

For our model, both the measures are relatively closer to each other for 4/6 datasets (∼2% difference) and somewhat apart (∼5% difference) for the other two datasets. For absent keyphrase generation (Tables 4 and 5), the differences are not as stark, considering the low values across all KP20K trained models. These observations indicate the need to study the evaluation measures more closely in future. Within the current setup, it is perhaps worthwhile to consider reporting F1@O along side F1@M though, since they both seem to consider different aspects for measurement.

Performance difference between KP20K and KPTimes For both present and absent keyphrases, we observe a superior performance with the KPTimes dataset compared to KP20K, across all the models. The difference is more stark for absent keyphrases, where the drop for our KPTimes trained model is 12% compared to present keyphrases, but close to 30% for KP20K trained model. Some qualitative analysis of our model output (examples are in Table A10) along with what we know about how the datasets are created lead us to two main observations:

1. The text part of KP20K consists of only abstracts, whereas KPTimes includes full text articles. Thus, in many cases, there just wasn't enough information in the abstract to generate the labeled abstractive keyphrases (even when those keyphrases were actually directly present in the later text). This could explain the low performance of absent keyphrase generation for KP20K, despite having close to 60% overlap between the ground truth absent keyphrases of train and test sets.

2. KP20K's keyphrases are author annotated, with no further verification. The annotation process for KPTimes was relatively more structured, where experienced editors added the keyphrases to the document by considering the tag suggestions from an automated labeler, adding new tags as needed. This could have resulted in a more consistent labeling of both present and absent keyphrases in KPTimes, which can explain the superior performance of all the models on this dataset.

A more detailed qualitative analysis of the datasets themselves, and the similarities and dissimilarities among the models can lead into further insights into this performance difference, and identifying potential coverage issues in the dataset labeling process in future.

6 Conclusions

We proposed to improve the keyphrase generation performance by a) developing a new approach for absent keyphrase generation with diversity heads and b) separating present keyphrase extraction as a plain sequence labeling problem. Our experiments with six standard datasets consisting of short as well as long documents showed that our present keyphrase generation model outperformed the state-of-the-art on four of the six datasets, and our absent keyphrase generation model outperformed the state-of-the-art on five of the six datasets. The insights from further analysis of results on the evaluation measures and the nature of the datasets point to interesting directions for future work. Improving cross-domain transfer, and exploring the portability of our approach to other languages are other possible future directions. It is important to note that the performance for this task reports much lower overall scores compared to other NLP tasks even with the recent generative LLMs like ChatGPT. This emphasizes the need for more focused research on the nature of the task itself along with model development.

7 Limitations

Our experiments in this paper have focused only on English datasets and supervised learning scenarios, which can be considered a limitation. Our model, while performing over the state-of-the-art for many datasets, still does poorly with cross-domain datasets (e.g., KP20K trained model tested on KPTimes or viceversa). Finally, our evaluation (which is the standard procedure for KPG) is more

of a surface comparison and not a semantic comparison of the ground truth and predicted output, thus ignoring the keyphrases that semantically closer but lexically different from ground truth. There are also potential coverage issues in the original labeled keyphrases in one of the datasets, as qualitative analysis revealed some instances where the model's labels seem appropriate but do not figure in the ground truth (examples in Table A10), which our evaluation process does not address at this point. Additionally, while we separate extraction and generation, the two tasks indeed share some commonalities, which the complete separation ignores. A better multi-task framing of the task, where extraction can learn from generation and vice-versa may result in a better performance. Finally, our experimental choices (e.g., number of epochs, not doing multiple runs of experiments) are limited by the compute availability, which can be considered a limitation in terms of exploring the experimental space in full. All our results and findings are to be understood in the context of these limitations.

8 Ethics Statement

The research did not involve in the creation of any data artifacts and used publicly accessible English datasets. However, some of the dataset instances can be directly traced to the original authors and writeups, through a simple google search. We share our code as supplementary material at `https://github.com/edwinthomas444/diverse-keyphrase-generation` for supporting reproducible research.

9 Acknowledgements

We thank the three anonymous reviewers for their constructive comments and active engagement during the author rebuttal phase, and we think the discussion eventually made the paper better than the initial version. We also thank Gabriel Bernier-Colborne and Taraka Rama for their comments on the initial draft. Finally, we thank the authors of the models we replicated, for sharing their code in a usable manner. This research was conducted at the National Research Council of Canada, thereby establishing a copyright belonging to the Crown in Right of Canada, that is, to the Government of Canada.

References

Wasi Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2021. Select, extract and generate: Neural keyphrase generation with layer-wise coverage attention. 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 1389–1404, Online. Association for Computational Linguistics.

Hareesh Bahuleyan and Layla El Asri. 2020. Diverse keyphrase generation with neural unlikelihood training. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5271–5287, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4057–4066, Brussels, Belgium. Association for Computational Linguistics.

Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019a. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2846–2856, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019b. Title-guided encoding for keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6268–6275.

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.

Md Faisal Mahbub Chowdhury, Gaetano Rossiello, Michael Glass, Nandana Mihindukulasooriya, and

Alfio Gliozzo. 2022. Applying a generic sequence-to-sequence model for simple and effective keyphrase generation. *arXiv preprint arXiv:2201.05302*.

A Diya and I Mizuho. 2022. Keyphrase generation by utilizing bart finetuning and bert-based ranking. In *DEIM Forum*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Ygor Gallina, Florian Boudin, and Béatrice Daille. 2019. Kptimes: A large-scale dataset for keyphrase generation on news documents. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 130–135.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.

Jihyuk Kim, Myeongho Jeong, Seungtaek Choi, and Seung-won Hwang. 2021. Structure-augmented keyphrase generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2657–2667, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Fajri Koto, Timothy Baldwin, and Jey Han Lau. 2022. LipKey: A large-scale news dataset for absent keyphrases generation and abstractive summarization. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3427–3437, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Tuhin Kundu, Jishnu Ray Chowdhury, and Cornelia Caragea. 2023. Neural keyphrase generation: Analysis and evaluation. *arXiv preprint arXiv:2304.13883*.

Giuseppe Lancioni, Saida S.Mohamed, Beatrice Portelli, Giuseppe Serra, and Carlo Tasso. 2020. Keyphrase generation with GANs in low-resources scenarios. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 89–96, Online. Association for Computational Linguistics.

Rui Liu, Zheng Lin, and Weiping Wang. 2021. Addressing extraction and generation separately: Keyphrase prediction with pre-trained language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3180–3191.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 135–144, Portland, Oregon, USA. Association for Computational Linguistics.

Yichao Luo, Yige Xu, Jiacheng Ye, Xipeng Qiu, and Qi Zhang. 2021. Keyphrase generation with fine-grained evaluation-guided reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 497–507, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. 2021. An empirical study on neural keyphrase generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4985–5007, Online. Association for Computational Linguistics.

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.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.

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.

Tokala Yaswanth Sri Sai Santosh, Nikhil Reddy Varimalla, Anoop Vallabhajosyula, Debarshi Kumar Sanyal, and Partha Pratim Das. 2021a. Hicova: Hierarchical conditional variational autoencoder for

keyphrase generation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3448–3452.

TYSS Santosh, Debarshi Kumar Sanyal, Plaban Kumar Bhowmick, and Partha Pratim Das. 2021b. Gazetteer-guided keyphrase generation from research papers. In *Advances in Knowledge Discovery and Data Mining: 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11–14, 2021, Proceedings, Part I*, pages 655–667. Springer.

Xianjie Shen, Yinghan Wang, Rui Meng, and Jingbo Shang. 2022. Unsupervised deep keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11303–11311.

Mingyang Song, Yi Feng, and Liping Jing. 2023a. A survey on recent advances in keyphrase extraction from pre-trained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2153–2164, Dubrovnik, Croatia. Association for Computational Linguistics.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R. Lyu, and Shuming Shi. 2019. Topic-aware neural keyphrase generation for social media language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2516–2526, Florence, Italy. Association for Computational Linguistics.

Yue Wang, Jing Li, Michael Lyu, and Irwin King. 2020. Cross-media keyphrase prediction: A unified framework with multi-modality multi-head attention and image wordings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3311–3324, Online. Association for Computational Linguistics.

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

Di Wu, Wasi Uddin Ahmad, and Kai-Wei Chang. 2023a. Rethinking model selection and decoding for keyphrase generation with pre-trained sequence-to-sequence models. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Di Wu, Wasi Uddin Ahmad, and Kai-Wei Chang. 2024. On leveraging encoder-only pre-trained language models for effective keyphrase generation. *arXiv preprint arXiv:2402.14052*.

Di Wu, Da Yin, and Kai-Wei Chang. 2023b. Kpeval: Towards fine-grained semantic-based evaluation of keyphrase extraction and generation systems. *arXiv preprint arXiv:2303.15422*.

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. 2022b. Fast and constrained absent keyphrase generation by prompt-based learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11495–11503.

Binbin Xie, Jia Song, Liangying Shao, Suhang Wu, Xiangpeng Wei, Baosong Yang, Huan Lin, Jun Xie, and Jinsong Su. 2023. From statistical methods to deep learning, automatic keyphrase prediction: A survey. *Information Processing & Management*, 60(4):103382.

Binbin Xie, Xiangpeng Wei, Baosong Yang, Huan Lin, Jun Xie, Xiaoli Wang, Min Zhang, and Jinsong Su. 2022. WR-One2Set: Towards well-calibrated keyphrase generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7283–7293, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Hamada M Zahera, Daniel Vollmers, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. 2022. Multpax: Keyphrase extraction using language models and knowledge graphs. In *The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings*, pages 303–318. Springer.

Yong Zhang and Weidong Xiao. 2018. Keyphrase generation based on deep seq2seq model. *IEEE access*, 6:46047–46057.

Guangzhen Zhao, Guoshun Yin, Peng Yang, and Yu Yao. 2022. Keyphrase generation via soft and hard semantic corrections. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7757–7768, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jing Zhao and Yuxiang Zhang. 2019. Incorporating linguistic constraints into keyphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5224–5233, Florence, Italy. Association for Computational Linguistics.

Cangqi Zhou, Jinling Shang, Jing Zhang, Qianmu Li, and Dianming Hu. 2021. Topic-attentive encoder-decoder with pre-trained language model for keyphrase generation. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1529–1534. IEEE.

A Appendix

Dataset Statistics More detailed dataset statistics are shown in Tables A1 and A2.

Detailed Performance Measures Tables A3 and A4 show additional evaluation measures (P/R/F@O,M,K where K=5) for both present and absent keyphrase extraction.

Effect of Diversity Heads: Tables A5 and A6 provide detailed results for Seq2Seq and Seq2Set (without diversity heads).

Beam Search Decoding: Table A7 shows the detailed results for using beam search (n=5) instead of the default greedy decoding for our model.

Comparisons between scaled models: Table A8 and Table A9 show the detailed results parameter scaled Seq2Set and TransSet models respectively, when trained with KP20K-TRAIN dataset.

Example Outputs Table A10 shows some example outputs from our model trained on KP20K dataset, illustrating some of the issues in the dataset and our model. For example, in the first example, the predicted extractive and abstractive keyphrases seem to suit the context, but the ground truth has too few keyphrases. In the second example, the ground truth abstractive phrase is difficult to infer from the provided abstract, and the model's outputs include incomplete words. In the third example, both model and ground truth agree with each other better, and in the fourth example, the outputs seem to represent the text better than the ground truth (especially abstractive).

Dataset	#docs	#words/doc	#kp	#uniquekp	avg len extractivekp	avg len abstractivekp	#presentkp/doc	#absentkp/doc
Train sets								
KP20k	∼530k	157.8	∼2.8m	∼723k	1.9	2.15	2.34	2.94
KPTimes	∼260k	783.32	∼1.3m	∼102k	1.62	2.5	2.15	2.88
Test sets								
KP20k	20k	157.94	∼105k	∼57k	1.9	2.15	2.34	2.93
KPTimes	20k	643.24	∼100k	∼21k	1.5	2.09	2.72	2.31
Inspec	500	134.6	∼5k	∼4.6k	2.21	2.49	6.57	3.26
Krapivin	2304	∼9k	∼12k	8728	1.96	2.26	3.73	1.6
SemEval	100	∼8k	∼1.5k	1388	1.95	2.4	9.2	6
NUS	211	∼8k	∼2k	∼2k	1.87	2.49	8	3.07

Table A1: More detailed statistics about the datasets used

Overlap-Extractive			Overlap-Abstractive		
	KP20k-Train	KPTimes-Train		KP20k-Train	KPTimes-Train
KP20K-Test	70.9	2.83	KP20K-Test	61.64	0.96
Inspec-Test	42.68	3.57	Inspec-Test	31.55	1.66
SemEval-Test	57.16	3.18	SemEval-Test	29.26	0.17
NUS-Test	69.07%	3.61	NUS-Test	47.39	0.63%
Krapivin-Test	75.83	1.46	Krapivin-Test	71.76	0.38
KPTimes-Test	18.15	51.89	KPTimes-Test	11.27	63.96

Table A2: Overlap between different train/test sets, separated by Extractive and Abstractive keyphrases (without stemming)

Train: KP20K									
test set	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kp20k	30.74	64.56	41.65	46.32	44.99	45.64	54.6	33.88	41.81
krapivin	27.72	57.22	37.35	39.99	41.76	40.85	50.09	32.37	39.33
inspec	37.02	28.4	32.14	42.83	20.52	27.75	41.38	21.85	28.6
semeval	32.57	50.47	39.59	59.21	34.13	43.3	56.25	31.65	40.5
nus	41.89	57.84	48.59	58.44	37.44	45.64	61.86	33.06	43.09
Train: KPTimes									
test set	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kptimes	44.99	75.34	56.34	60.3	58.94	59.61	70.78	46.97	56.47
krapivin	9.31	5.94	7.25	9.31	5.62	7.01	10.25	5.51	7.17
inspec	13.14	4.06	6.21	13.47	3.65	5.75	13.2	3.69	5.76
semeval	30.56	16.95	21.81	33.33	14.71	20.41	33.08	12.99	18.65
nus	26.56	10.36	14.91	28.87	9.85	14.69	28.29	9.68	14.42

Table A3: Performance of our model for Present Keyphrase Extraction

Train: KP20K									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kp20k	6.87	9.11	7.84	6.87	9.11	7.84	7.48	6.4	6.9
krapivin	7.03	8.25	7.59	7.03	8.25	7.59	7.95	6.96	7.42
inspec	1.21	1.16	1.18	1.21	1.16	1.18	1.04	0.77	0.89
semeval	9.04	2.74	4.21	9.04	2.74	4.21	9.04	2.74	4.21
nus	7.98	7.48	7.72	7.98	7.48	7.72	9.05	5.47	6.82
Train: KPTimes									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kptimes	43.07	45.22	44.12	43.07	45.22	44.12	45.27	37.76	41.18
krapivin	0.14	0.23	0.17	0.14	0.23	0.17	0.12	0.12	0.12
inspec	0.22	0.17	0.19	0.22	0.17	0.19	0.22	0.17	0.19
semeval	0	0	0	0	0	0	0	0	0
nus	0.8	0.31	0.45	0.8	0.31	0.45	0.8	0.31	0.45

Table A4: Performance of our model for Absent Keyphrase Generation

Train:KP20K									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kp20k	8.83	4.84	6.25	8.84	4.84	6.25	8.87	4.7	6.14
krapivin	10.62	3.97	5.78	10.62	3.97	5.78	10.62	3.97	5.78
inspec	1.58	1.2	1.36	1.58	1.2	1.36	1.86	1.2	1.46
semeval	11.62	1.69	2.95	11.62	1.69	2.95	11.62	1.69	2.95
nus	8.29	2.63	4.0	8.29	2.63	4.0	8.54	2.63	4.02
Train:KPTimes									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kptimes	45.0	38.05	41.24	45.05	38.02	41.24	46.55	34.62	39.71
krapivin	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
inspec	0.39	0.43	0.41	0.39	0.43	0.41	0.3	0.17	0.22
semeval	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
nus	0.75	0.16	0.27	0.75	0.16	0.27	0.75	0.16	0.27

Table A5: Seq2Seq Model Detailed Results for AKP (Row 1 in Table 6)

Train:KP20K									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kp20k	6.81	5.89	6.31	6.81	5.89	6.31	7.11	4.91	5.81
krapivin	7.48	6.78	7.11	7.48	6.78	7.11	8.06	6.08	6.93
inspec	0.95	1.04	0.99	0.95	1.04	0.99	1.21	1.04	1.12
semeval	10.02	2.75	4.31	10.02	2.75	4.31	10.27	2.75	4.33
nus	6.57	3.98	4.96	6.57	3.98	4.96	6.49	3.23	4.31
Train:KPTimes									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kptimes	49.15	29.64	36.98	49.15	29.64	36.98	49.15	27.78	35.5
krapivin	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
inspec	0.13	0.04	0.06	0.13	0.04	0.06	0.13	0.04	0.06
semeval	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
nus	1.01	0.31	0.47	1.01	0.31	0.47	1.01	0.31	0.47

Table A6: Seq2Set Model Detailed Results for AKP (Row 2 in Table 6)

Train:KP20K									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kp20k	8.25	11.71	9.68	8.25	11.71	9.68	10.02	8.86	9.4
krapivin	6.73	10.32	8.15	6.73	10.32	8.15	8.38	7.62	7.98
inspec	1.74	2.7	2.11	1.74	2.7	2.11	2.13	1.92	2.02
semeval	9.14	3.25	4.79	9.14	3.25	4.79	9.14	3.25	4.79
nus	8.78	7.65	8.18	8.78	7.65	8.18	9.05	5.73	7.02
Train:KPTimes									
dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kptimes	39.59	47.68	43.26	39.59	47.68	43.26	46.66	41.86	44.13
krapivin	0.14	0.23	0.17	0.14	0.23	0.17	0	0	0
inspec	0.35	0.28	0.32	0.35	0.28	0.32	0.35	0.28	0.32
semeval	0	0	0	0	0	0	0	0	0
nus	0.96	0.38	0.54	0.96	0.38	0.54	0.96	0.38	0.54

Table A7: Performance of our model with beam search decoding (n=5)

dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kp20k	11.1	4.61	6.51	11.1	4.61	6.51	11.11	4.56	6.47
krapivin	11.9	4.72	6.76	11.9	4.72	6.76	12.02	4.72	6.78
inspec	1.17	0.93	1.04	1.17	0.93	1.04	1.17	0.93	1.04
semeval	11.62	1.53	2.7	11.62	1.53	2.7	11.62	1.53	2.7
nus	11.81	2.66	4.35	11.81	2.66	4.35	11.81	2.66	4.35

Table A8: Detailed results for the scaled Seq2Set model (trained on KP20K-TRAIN)

dataset	P@M	R@M	F1@M	P@K	R@K	F1@K	P@O	R@O	F1@O
kp20k	4.99	4.35	4.65	4.99	4.35	4.65	5.01	3.52	4.13
krapivin	6.92	4.73	5.62	6.92	4.73	5.62	6.9	4.11	5.15
inspec	2.11	1.87	1.98	2.11	1.87	1.99	2.15	1.48	1.74
semeval	8.12	2.04	3.27	8.12	2.04	3.27	8.12	2.04	3.27
nus	6.75	4.1	5.1	6.75	4.1	5.1	6.35	2.68	3.77

Table A9: Detailed results for the scaled TransSet model (trained on KP20K-TRAIN)

a framework to automate the parsing of arabic language sentences . this paper proposes a framework to automate the parsing (sic) of arabic language sentences in general , although it focuses on the simple verbal sentences but it can be extended to any arabic language sentence . the proposed system is divided into two separated phases which are lexical analysis and syntax analysis . lexical phase analyses the words , finds its originals and roots , separates it from prefixes and suffixes , and assigns the filtered words to special tokens . syntax analysis receives all the tokens and finds the best grammar for the given sequence of the tokens by using context free grammar . our system assumes that the entered sentences are correct lexically and grammatically .
Ground truth-extractive: lexical analysis ; syntax analysis
Pred-extractive: syntax analysis ; arabic language sentences ; lexical analysis ; parsing ; context free grammar
Ground truth-abstractive: arabic language parser
Pred-abstractive: arabic language ; natural language processing ; lexical context ; lexical parsing ; sentence parsing

existence of solutions of abstract fractional integrodifferential equations of Sobolev type This paper deals with the study of existence of solutions of nonlinear fractional integrodifferential equations of Sobolev type with nonlocal condition in Banach spaces. The results are obtained by using resolvent operators, fractional calculus and fixed point technique. An example is provided to illustrate the theory.
Ground truth - extractive: sobolev type ; resolvent operators ; fractional integrodifferential equations
Pred-extractive: sobolev type ; resolvent operators ; fractional integrodifferential equations ; fractional calculus ; existence ; fixed point technique
Ground truth - abstractive: krasnoselskii fixed point theorem
Pred-Abstractive: fixed point theorem ; nonlocal integ ; resolvent integro

an action compiler targeting standard ml . we present an action compiler that can be used in connection with an action semantics based compiler generator . our action compiler produces code with faster execution times than code produced by other action compilers , and for some nontrivial test examples it is only a factor of two slower than the code produced by the gnu c compiler . targeting standard ml makes the description of the code generation simple and easy to implement . the action compiler has been tested on a description of the core of standard ml and a subset of c .
Ground truth - extractive: code generation ; action semantics ; standard ml
Pred-extractive: action compiler ; action semantics ; standard ml
Ground truth-abstractive: compiler generation
Pred-abstractive: compilation ; compiler generator

exploiting discourse information to identify paraphrases . we show the relation between discourse units and paraphrasing . we propose a new method for computing text similarity based on elementary discourse units . we apply the method to the task of paraphrase identification . we achieved < digit > . 4 accuracy in experiments conducted on the pan corpus . **Ground truth - extractive**: elementary discourse unit ; text similarity ; paraphrase identification
Pred-extractive: elementary ; discourse ; para ; paraphrase identification ; pan corpus ; discourse unit ; text similarity
Ground truth-abstractive: support vector machine ; mt metric ; discourse segmentation
Pred-abstractive: discourse analysis ; natural language processing ; paraphrase extraction

Table A10: Examples of Model Outputs - KP20K