# KPDrop: Improving Absent Keyphrase Generation

Jishnu Ray Chowdhury♠ Seoyeon Park♠ * Tuhin Kundu♣ * † Cornelia Caragea♠
♠ Computer Science, University of Illinois at Chicago ♣Amazon
{jraych2,spark313,cornelia}@uic.edu
tuhinkundu@outlook.com

## Abstract

Keyphrase generation is the task of generating phrases (keyphrases) that summarize the main topics of a given document. Keyphrases can be either present or absent from the given document. While the extraction of present keyphrases has received much attention in the past, only recently a stronger focus has been placed on the generation of absent keyphrases. However, generating absent keyphrases is challenging; even the best methods show only a modest degree of success. In this paper, we propose a model-agnostic approach called keyphrase dropout (or KPDrop) to improve *absent keyphrase generation*. In this approach, we randomly drop present keyphrases from the document and turn them into artificial absent keyphrases during training. We test our approach extensively and show that it consistently improves the absent performance of strong baselines in both supervised and resource-constrained semi-supervised settings[1].

## 1 Introduction

Keyphrase generation (KG) is the task of producing a set of phrases that best summarize a document. It can be leveraged for various applications such as text summarization (Zhang et al., 2017), recommendation (Bai et al., 2018), and opinion mining (Meng et al., 2012). Accurate identification of keyphrases especially on scientific papers can improve efficiency in paper indexing and paper retrieval (Chen et al., 2019a; Boudin et al., 2020).

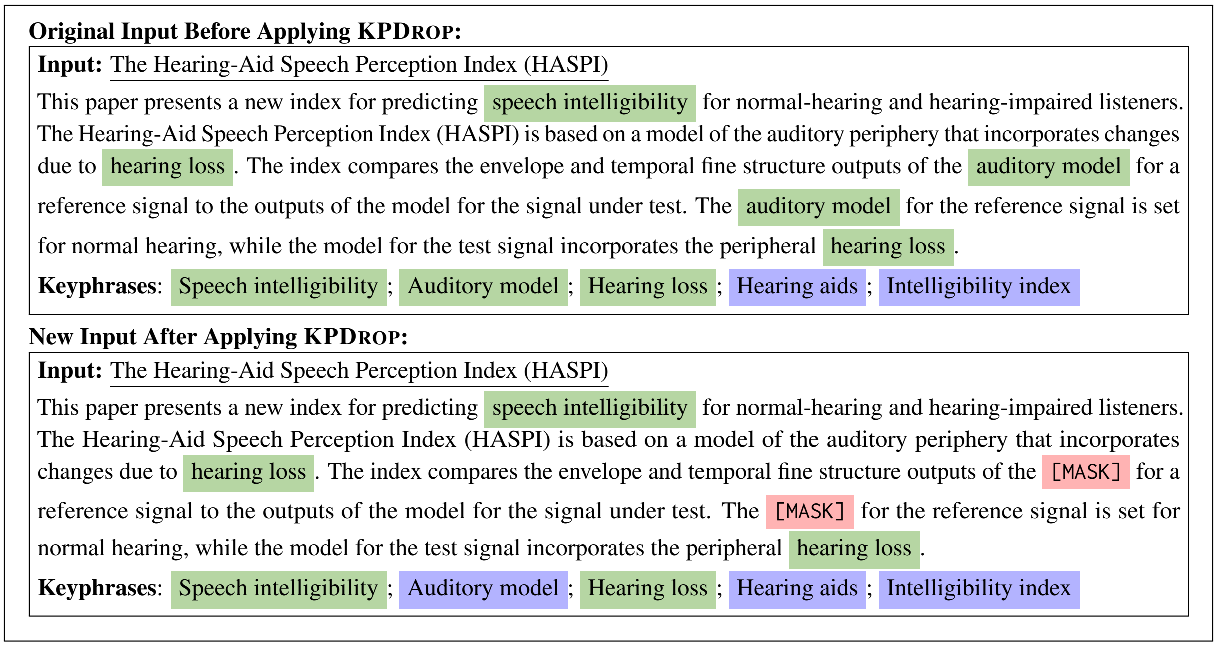
Keyphrases can be divided into two types: (1) *present* keyphrases and (2) *absent* keyphrases. Present keyphrases appear verbatim in the document, whereas absent keyphrases are topically-relevant but missing from the document. Many prior works (Hasan and Ng, 2014; Ünlü and Çetin, 2019) focused on present keyphrase *extraction* exclusively. As such, they are not suitable for generating any absent keyphrases. To overcome this limitation, recent approaches (Meng et al., 2017; Yuan et al., 2020; Chen et al., 2020; Ye et al., 2021b) use sequence-to-sequence (seq2seq) models to generate both present and absent keyphrases.

As shown by Boudin and Gallina (2021), absent keyphrases that are substantially different from the present ones can significantly improve the effectiveness of document-retrieval. Boudin and Gallina (2021) suggested that absent keyphrases can improve document-retrieval by expanding the query terms to alleviate the vocabulary mismatch problem between the query terms and relevant documents (Furnas et al., 1987). Thus, there is a strong motivation for improving absent keyphrase performance. However, generating absent keyphrases can be very challenging. Even the best keyphrase generation models (Chen et al., 2020; Ye et al., 2021a,b) still only achieve a modest performance in absent keyphrases.

In this work, we propose *keyphrase dropout* or KPDrop as a simple and effective technique to improve the performance of *absent* keyphrases. Unlike the traditional dropout method that focuses on dropping neurons (Srivastava et al., 2014), we propose a novel dropout method where, instead of neurons, we randomly drop entire phrases (specifically, present gold keyphrases) from a given document during training. Thus, the dropped present keyphrases are turned into (artificial) absent keyphrases. As a result, KPDrop has the following two effects:

1. The model is forced to deeply utilize the context information to infer dropped keyphrases that could have been otherwise simply extracted from the text. Thereby, the capability to infer missing keyphrases in general (including naturally missing ones) is increased.

* Equal contribution
† Work done at the University of Illinois at Chicago before Amazon
[1]Code: https://github.com/JRC1995/KPDrop



**Original Input Before Applying KPDrop:**

**Input:** The Hearing-Aid Speech Perception Index (HASPI)

This paper presents a new index for predicting speech intelligibility for normal-hearing and hearing-impaired listeners. The Hearing-Aid Speech Perception Index (HASPI) is based on a model of the auditory periphery that incorporates changes due to hearing loss. The index compares the envelope and temporal fine structure outputs of the auditory model for a reference signal to the outputs of the model for the signal under test. The auditory model for the reference signal is set for normal hearing, while the model for the test signal incorporates the peripheral hearing loss.

**Keyphrases**: Speech intelligibility ; Auditory model ; Hearing loss ; Hearing aids ; Intelligibility index

**New Input After Applying KPDrop:**

**Input:** The Hearing-Aid Speech Perception Index (HASPI)

This paper presents a new index for predicting speech intelligibility for normal-hearing and hearing-impaired listeners. The Hearing-Aid Speech Perception Index (HASPI) is based on a model of the auditory periphery that incorporates changes due to hearing loss. The index compares the envelope and temporal fine structure outputs of the [MASK] for a reference signal to the outputs of the model for the signal under test. The [MASK] for the reference signal is set for normal hearing, while the model for the test signal incorporates the peripheral hearing loss.

**Keyphrases**: Speech intelligibility ; Auditory model ; Hearing loss ; Hearing aids ; Intelligibility index

Figure 1: An example of how the input document (Kates and Arehart, 2014), and the keyphrases change after applying KPDrop. Here, KPDrop drops the present keyphrase "auditory model". Green highlighting indicates present keyphrases, blue highlighting indicates absent keyphrases, and red highlighting indicates mask tokens.

2. The method stands as a dynamic data augmentation strategy. KPDrop can be used to modify a given set of training documents by randomly dropping some present keyphrases from the documents turning them into artificial absent keyphrases. As such, new data can be created from the originals just like a standard data augmentation technique. Moreover, the process can be dynamically done in real-time during training. Thus, in different epochs different present keyphrases may be dropped for the same document yielding higher diversity in the data for model training.

We apply KPDrop into three distinct neural models representing three distinct paradigms: one2many (Yuan et al., 2020), one2one (Meng et al., 2017; Huang et al., 2021), and one2set (Ye et al., 2021b). We observe consistent and substantial improvement in absent keyphrase performance in supervised settings on five standard datasets used for KG evaluation, with little to no drops in present keyphrase performance (and in fact often yielding improved present performance). Additionally, KPDrop can be used to create synthetic absent keyphrases for unlabelled data to be used for self-supervised pre-training. We demonstrate that such pre-training augmented with KPDrop brings substantial improvement when fine-tuned in low-resource labelled data.

## 2 Methodology of KPDrop

In this section, we formally describe our approach of Keyphrase Dropout (KPDrop). In keyphrase generation (KG), we have a document $X$ as an input sequence of tokens, and a set of $n$ keyphrases, $Y = \{y_1, y_2, \ldots, y_n\}$, as the target output. Within $Y$, there is a subset of $s$ ($0 \leq s \leq n$) present keyphrases as $P = \{p_1, p_2, p_3, \ldots, p_s\}$ ($P \subseteq Y$) and a subset of $t$ ($0 \leq t \leq n$) absent keyphrases as $A = \{a_1, a_2, a_3, \ldots, a_t\}$ ($A \subseteq Y$). It also holds true that $A \cap P = \emptyset$ and $s + t = n$. Similar to Meng et al. (2017), we separate absent and present keyphrases by checking if the stemmed version of a keyphrase appears in the stemmed version of the input document (in which case it is present) or does not appear (in which case it is absent).

When applying KPDrop, any present keyphrase $p_k \in P$ can be randomly dropped with a probability of $r$ (sampled from binomial distribution), where $r$ is the dropout rate (or KPDrop rate). Let $O$ ($O \subseteq P$) be the subset of present keyphrases that are randomly dropped during the application of KPDrop at some training epoch. For every keyphrase $p_k \in O$, we remove *any and all* substrings from $X$ that, when stemmed, match the stemmed version of the phrase $p_k$ and replace each removed substring with a special mask token [MASK]. We call the modified version of $X$ as $X^{new}$. The sets of present and absent keyphrases are also modified. The new set of present keyphrases becomes $P := P \setminus O$ and the new set of absent keyphrases becomes $A := A \cup O$. Thus, the keyphrases in $O$, which were originally present, become artificially absent after applying KPDROP. The set $Y$ becomes $Y^{new}$ with the new versions of $P$ and $A$. An example of applying KPDROP is shown in Figure 1.

Note that although as a set $Y^{new}$ is the same as $Y$, we use $Y^{new}$ to convey that the sets of present and absent keyphrases have changed.

### 2.1 Two Strategies of Applying KPDROP

KPDROP can be applied to a given mini-batch of examples in at least two distinct ways which act as data augmentation (noising) strategies.

**KPDROP-R:** For the first strategy that we refer to as KPDROP-REPLACE or KPDROP-R, we can think of KPDROP as a dropout technique and like other applications of dropout we can choose to *replace* the original examples with their corresponding keyphrase-dropped versions. In other words, each original sample $(X, Y)$ in a mini-batch $B$ is replaced with $(X^{new}, Y^{new})$ obtained after applying KPDROP. More formally, if initially we had a mini-batch set $B$, after applying KPDROP-R, we get a new batch $B_{KPDR}$ as $B_{KPDR} = \{(X^{new}, Y^{new}) | (X, Y) \in B\}$. Note that at any given training iteration, we create a *single* KPDropped counterpart $(X^{new}, Y^{new})$ for each sample $(X, Y)$ in the mini-batch.

**KPDROP-A:** For the second strategy that we refer to as KPDROP-APPEND or KPDROP-A, we can think of KPDROP as closer to a data augmentation technique where instead of replacing the original examples we *augment* the original batch with the new keyphrase dropped versions of the originals. In other words, for each original sample $(X, Y)$ in a mini-batch, we add $(X^{new}, Y^{new})$ obtained after applying KPDROP. More formally, starting with the same mini-batch set $B$ as before, after applying KPDROP-A, we get a new batch $B_{KPDA}$ as $B_{KPDA} = B \cup B_{KPDR}$.

Given that KPDROP-R can increase the number of absent keyphrases per sample during training, we hypothesize that the model can learn to be more biased towards generating absent keyphrases. This can help to increase absent keyphrase performance at the cost of present keyphrase performance because the technique will be dropping present keyphrases. On the other hand, KPDROP-A, instead of replacing the original data, it can offer extra samples per batch that have additional artificially absent keyphrases. Thus, KPDROP-A should be still able to improve absent keyphrase performance while also maintaining the original data with its original present keyphrases. Thus, KPDROP-A will offer the underlying model with the same opportunity to learn present keyphrases as would the model without KPDROP-A. Intuitively, this can help improve absent performance without a substantial cost to present performance.

### 2.2 KPDROP Features and Connections to Works from Other Areas

In this section, we highlight some notable features of KPDROP and draw connections to some relevant works outside the area of keyphrase generation.

**KPDROP and Masked Language Modeling:** Superficially, KPDROP is similar to a standard masked language modeling (MLM) (Devlin et al., 2019; Raffel et al., 2020) strategy insofar that both involve the reconstruction of some masked out text-spans. However, there are several crucial differences between KPDROP and vanilla MLM. First, KPDROP masks only *present keyphrases*, not random subspans or phrases. Second, KPDROP masks *all instances* of the selected present keyphrases in the document to make them truly absent (see Figure 1). In contrast, MLM does not mask *all instances* of the masked token or phrase. Third, masking (replacing a subspan with a mask token) is only an engineering choice for KPDROP; not an essential ingredient. We can simply drop the selected present keyphrases without replacing them with a mask token. In KPDROP, predictions of artificial absent keyphrases (masked present keyphrases) are not *explicitly* associated with any mask position.

**KPDROP as Structured Dropout:** KPDROP is a unique kind of *structured dropout* where all instances of some randomly chosen present keyphrases are dropped. There are other examples of structured dropout in prior works. For example, Iyyer et al. (2015) proposed word dropout (for general NLP tasks) where whole word embeddings are dropped instead of just random neurons. Huang et al. (2016) and Fan et al. (2020) used another form of structured dropout that stochastically drops whole layers for general computer vision tasks and for general NLP tasks respectively.

**Model-Agnosticism of KPDROP:** One important feature of KPDROP is that it is model-agnostic. KPDROP only requires changing the input and output without making internal architectural changes. As such, it is compatible with any model that is suitable for KG. KPDROP can also be easily stacked with other techniques that help absent keyphrase generation. Later (in §5.1), we demonstrate that KPDROP works just as well for very different architectures: (1) an RNN-based model trained in one2many settings (Yuan et al., 2020) (2) an RNN-based model trained in one2one settings (Meng et al., 2017; Huang et al., 2021), and (3) a Transformer-based model trained in one2set settings (Ye et al., 2021b).

## 3 Evaluations

We use the following evaluation metrics:

**$F_1$@M**: $F_1$@M is a standard metric (Yuan et al., 2020; Chen et al., 2019b; Chan et al., 2019; Chen et al., 2020; Ye et al., 2021b) used to evaluate the performance of keyphrase generation. This is an $F_1$ based metric where *all* the predictions by a given model are considered for evaluation.

**$F_1$@5**: $F_1$@5 is similar to $F_1$@M but only the top 5 predicted keyphrases are used for evaluation (if the total number of predictions are less than 5, all the available predictions are used). This metric is used in several prior works (Meng et al., 2017; Yuan et al., 2020).[2] This metric is useful in settings where the model is used to overgenerate keyphrases (for example, by using beam search). Overgeneration can lead to lower precision when all the predictions are kept (thus, low $F_1$@M). So, often in such settings, it is more useful to check the performance of the model when some simple truncation policy is used, for example, selecting some top $k$ (e.g., top 5 in $F_1$@5) predictions.

**R@10 and R@50**: R@10 and R@50 represents macro recall of the top 10 predictions and the top 50 predictions, respectively. In some applications (such as retrieval) high recall can be sometimes more useful than precision. R@10 and R@50 are used in prior works (Meng et al., 2017; Chen et al., 2018; Liu et al., 2020) to measure absent keyphrase performance after overgenerating them using beam search with high beam size.

Like prior works, we use macro-$F_1$ and macro-recall for all the above metrics.

## 4 Baselines

We use the following baselines with our KPDROP-R and KPDROP-A approaches:

**GRU One2Many**: GRU One2Many (also known as CatSeq) represents a simple seq2seq baseline based on GRUs. It takes a document input and generates a concatenated sequence of keyphrases similar to Yuan et al. (2020). For this baseline, we concatenate the ground truth keyphrases based on the best performing ordering procedure (Meng et al., 2021): we first concatenate the present keyphrases according to the order of their first appearance in text and then we append the absent keyphrases in their natural order. However, when using KPDROP, we start with the ordering as mentioned but then shift the dropped present keyphrases (artificial absent keyphrases) after the fully present keyphrases but keep them before the naturally absent keyphrases. We maintain the internal order of the dropped present (artificially absent) keyphrases. We use ";" as the delimiter for separating keyphrases.

**GRU One2One**: GRU One2One (also known as CopyRNN) is another seq2seq model based on GRUs. However, unlike the One2Many model, it can predict only one keyphrase per generated sequence. It can be still used to generate multiple keyphrases by using beam search and preserving multiple beams of sequences (each representing a keyphrase). The One2One approach was first introduced by Meng et al. (2017) where the original training data was divided such that each input was associated with only one ground truth keyphrase. However, instead of that, we use the more efficient training approach of using reset states as in Huang et al. (2021). That is, we simply train the one2one model similar to one2many models using teacher forcing but we "reset" the hidden state when generating the first word of any ground truth keyphrase. Resetting (Huang et al., 2021) corresponds to using the initial RNN hidden state and the first special input token indicating start of sequence and thereby removing dependencies from previous generations.

**Transformer One2Set**: Transformer One2Set (also known as SetTrans) is a Transformer-based

[2]Note that the $F_1$@5 metric as used by us and as introduced by Meng et al. (2017) for keyphrase generation is different from the $F_1$@5 metric as introduced by Chan et al. (2019) and used in some other works (Chen et al., 2020; Ye et al., 2021b). We report results with Chan et al. (2019)'s formulation of $F_1$@5 in Appendix A.2

| Models | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 |
| **GRU One2Many** | | | | | | | | | | |
| Greedy | $1.4_2$ | $1.4_2$ | $2.9_6$ | $2.9_6$ | $3.4_4$ | $3.4_4$ | $2.4_2$ | $2.4_2$ | $3.3_3$ | $3.3_3$ |
| Greedy+KPD-R | $1.2_1$ | $1.2_1$ | $3.2_7$ | $3.2_7$ | $\mathbf{5.1_2}$ | $\mathbf{5.1_2}$ | $\mathbf{2.7_2}$ | $\mathbf{2.7_2}$ | $\mathbf{4.0_1}$ | $\mathbf{4.0_1}$ |
| Greedy+KPD-A | $\mathbf{1.5_1}$ | $\mathbf{1.5_1}$ | $\mathbf{3.5_6}$ | $\mathbf{3.5_6}$ | $4.6_3$ | $4.6_3$ | $2.6_4$ | $2.6_4$ | $3.9_3$ | $3.9_3$ |
| Beam | $2.8_1$ | $\mathbf{2.9_1}$ | $5.5_3$ | $5.5_2$ | $7.2_3$ | $7.3_4$ | $3.8_2$ | $3.8_2$ | $5.8_0$ | $5.8_1$ |
| Beam+KPD-R | $2.8_3$ | $\mathbf{2.9_3}$ | $6.5_2$ | $\mathbf{6.5_2}$ | $\mathbf{7.3_3}$ | $\mathbf{7.4_3}$ | $\mathbf{5.1_2}$ | $4.8_2$ | $6.1_0$ | $6.3_0$ |
| Beam+KPD-A | $\mathbf{3.0_1}$ | $\mathbf{2.9_1}$ | $\mathbf{6.6_{10}}$ | $\mathbf{6.5_9}$ | $\mathbf{7.3_3}$ | $7.3_4$ | $5.0_3$ | $\mathbf{5.0_4}$ | $\mathbf{6.4_0}$ | $\mathbf{6.5_0}$ |
| **GRU One2One** | | | | | | | | | | |
| Beam | $0.6_0$ | $2.8_1$ | $1.5_1$ | $5.7_7$ | $1.2_0$ | $5.9_3$ | $1.3_0$ | $3.8_1$ | $1.0_0$ | $6.2_0$ |
| Beam+KPD-R | $\mathbf{0.7_0}$ | $\mathbf{2.9_3}$ | $1.5_0$ | $\mathbf{7.5_3}$ | $\mathbf{1.3_0}$ | $\mathbf{7.8_2}$ | $1.4_0$ | $\mathbf{4.9_5}$ | $1.0_0$ | $6.5_1$ |
| Beam+KPD-A | $\mathbf{0.7_0}$ | $2.7_1$ | $\mathbf{1.6_0}$ | $6.7_2$ | $\mathbf{1.3_0}$ | $6.5_3$ | $\mathbf{1.5_1}$ | $4.1_3$ | $\mathbf{1.1_0}$ | $\mathbf{6.8_0}$ |
| **Transformer One2Set** | | | | | | | | | | |
| Greedy | $2.8_4$ | $2.8_4$ | $6.4_8$ | $6.4_8$ | $6.8_2$ | $6.8_2$ | $3.5_1$ | $3.5_1$ | $5.6_0$ | $5.5_0$ |
| Greedy+KPD-R | $2.9_1$ | $2.8_1$ | $6.9_8$ | $6.9_7$ | $\mathbf{8.4_8}$ | $\mathbf{8.4_8}$ | $4.6_3$ | $\mathbf{4.6_4}$ | $6.4_1$ | $6.3_1$ |
| Greedy+KPD-A | $\mathbf{3.2_2}$ | $\mathbf{3.2_2}$ | $\mathbf{7.4_9}$ | $\mathbf{7.4_9}$ | $7.2_7$ | $7.2_7$ | $\mathbf{4.7_1}$ | $\mathbf{4.6_1}$ | $\mathbf{6.6_1}$ | $\mathbf{6.5_1}$ |
| Beam | $0.4_0$ | $3.3_2$ | $0.9_0$ | $7.0_5$ | $0.8_0$ | $6.7_5$ | $0.8_0$ | $4.7_4$ | $\mathbf{0.6_0}$ | $5.8_0$ |
| Beam+KPD-R | $0.4_0$ | $2.4_1$ | $\mathbf{1.0_0}$ | $7.2_5$ | $\mathbf{0.9_0}$ | $7.3_2$ | $0.9_0$ | $5.2_4$ | $\mathbf{0.6_0}$ | $6.1_0$ |
| Beam+KPD-A | $\mathbf{0.5_0}$ | $\mathbf{3.6_2}$ | $\mathbf{1.0_0}$ | $\mathbf{7.9_4}$ | $\mathbf{0.9_0}$ | $\mathbf{7.8_2}$ | $\mathbf{1.0_0}$ | $\mathbf{5.3_4}$ | $\mathbf{0.6_0}$ | $\mathbf{6.7_0}$ |

Table 1: Absent keyphrase performance (**F1**) for different models. KPD represents KPDrop. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$). We bold the best scores per block.

model trained in a new One2Set paradigm as introduced by Ye et al. (2021b). In this method, a fixed number of preset control codes are used to generate all present and absent keyphrases in parallel independent of each other. Moreover, during training target keyphrases are matched with the predicted ones using Hungarian algorithm (Kuhn, 1955) so that the training is not sensitive to the order of the target keyphrases. Note that Transformer One2Set uses specialized control codes for present and absent keyphrase generation separately. Thus, when using KPDROP, we make sure to use the control codes for absent keyphrases to generate artificial absent keyphrases (dropped present keyphrases) as well.

## 5 Supervised Experiments

In the supervised setting, we explore the effects of applying both KPDROP-R and KPDROP-A on the baselines that we discussed in §4. Note that One2Many models (Yuan et al., 2020; Meng et al., 2021) had been explored in both greedy search based generations where only a single concatenated sequence of keyphrases is generated and also in beam search based generations where multiple beams of concatenated sequence of keyphrases are generated. We too explore both greedy and beam search for the One2Many models. In One2One models, greedy search is ineffective because it only generates a single keyphrase. Thus, following Meng et al. (2017), we only use beam search to overgenerate multiple beams of keyphrases for One2One models. For One2Set models, only greedy search was explored (Ye et al., 2021b). Here, in addition to greedy search, we also explore using beam search for each control code in One2Set models. For all models, we investigate the effect of applying KPDROP in all of their applicable decoding settings (be it beam search or greedy search). Following prior works (Meng et al., 2017; Chen et al., 2018), we mainly use beam search to demonstrate recall performance (Table 2) for absent keyphrase performance. The recall performance stands out best when using beam search. Greedy search can be more conservative with generation; thus, has limited recall. We evaluate our three baselines on five scientific datasets: KP20k (Meng et al., 2017), Inspec (Hulth, 2003), Krapivin (Krapivin et al., 2009), SemEval (Kim et al., 2010) and NUS (Nguyen and Kan, 2007). Following previous works, we use the training set of KP20k to train all models. All models are run three times on different seeds. We report the mean and standard deviation of these three runs. Hyperparameters are detailed in Appendix A.1.

### 5.1 Results on Absent Keyphrase Generation

In Table 1, we show the F1 performance for absent keyphrase generation. As we can see from the ta-

|  | Inspec |  | NUS |  | Krapivin |  | SemEval |  | KP20k |  |
|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **R@10** | **R@50** | **R@10** | **R@50** | **R@10** | **R@50** | **R@10** | **R@50** | **R@10** | **R@50** |
| **GRU One2Many** |  |  |  |  |  |  |  |  |  |  |
| Beam | $3.8_1$ | $3.8_1$ | $5.3_2$ | $5.3_2$ | $8.2_3$ | $8.2_3$ | $3.0_0$ | $3.0_0$ | $8.1_1$ | $8.1_1$ |
| Beam+KPD-R | $\mathbf{4.3_3}$ | $\mathbf{4.3_4}$ | $\mathbf{7.5_4}$ | $\mathbf{7.6_3}$ | $\mathbf{10.9_1}$ | $\mathbf{11.1_2}$ | $\mathbf{4.2_3}$ | $\mathbf{4.2_3}$ | $\mathbf{10.1_1}$ | $\mathbf{10.2_1}$ |
| Beam+KPD-A | $4.2_2$ | $4.2_2$ | $6.4_{10}$ | $6.4_{10}$ | $9.0_3$ | $9.1_3$ | $3.9_2$ | $4.0_2$ | $9.2_1$ | $9.3_1$ |
| **GRU One2One** |  |  |  |  |  |  |  |  |  |  |
| Beam | $6.1_3$ | $11.3_2$ | $9.5_{10}$ | $17.4_5$ | $12.3_2$ | $22.9_1$ | $4.4_2$ | $8.1_7$ | $14.0_0$ | $23.5_1$ |
| Beam+KPD-R | $\mathbf{6.3_3}$ | $12.2_4$ | $\mathbf{12.1_9}$ | $\mathbf{20.3_4}$ | $\mathbf{15.6_9}$ | $\mathbf{26.8_6}$ | $\mathbf{5.6_3}$ | $\mathbf{9.4_3}$ | $14.6_1$ | $24.8_3$ |
| Beam+KPD-A | $5.9_4$ | $\mathbf{12.9_7}$ | $11.3_7$ | $19.0_5$ | $14.5_7$ | $24.7_5$ | $5.0_4$ | $9.3_3$ | $\mathbf{15.2_1}$ | $\mathbf{25.5_1}$ |
| **Transformer One2Set** |  |  |  |  |  |  |  |  |  |  |
| Beam | $6.7_3$ | $12.5_3$ | $12.0_5$ | $19.6_3$ | $13.6_{10}$ | $24.4_9$ | $5.4_2$ | $9.1_2$ | $13.5_2$ | $23.4_2$ |
| Beam+KPD-R | $5.7_1$ | $11.7_3$ | $13.3_6$ | $21.2_5$ | $\mathbf{15.6_5}$ | $27.3_9$ | $\mathbf{5.9_8}$ | $10.4_5$ | $14.5_0$ | $25.4_2$ |
| Beam+KPD-A | $\mathbf{7.9_3}$ | $\mathbf{13.6_4}$ | $\mathbf{14.0_2}$ | $\mathbf{22.3_8}$ | $15.2_4$ | $\mathbf{27.7_4}$ | $\mathbf{5.9_5}$ | $\mathbf{11.0_7}$ | $\mathbf{15.6_2}$ | $\mathbf{26.6_2}$ |

Table 2: Absent keyphrase performance (**Recall**) for different models. KPD represents KPDrop. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$). We bold the best scores per block.

ble, KPDROP always boosts the absent keyphrase performance against a comparable baseline (regardless of whether we use KPDROP-R or KPDROP-A). Comparing greedy with beam search in GRU One2Many, we find that overgenerating with beam search can substantially improve the absent performance over greedy and further applying KPDROP to beam search improves the performance significantly (e.g., on KP20K performance improves from 5.8% to 6.5%). In both GRU One2One and Transformer One2Set, we observe that beam search leads to overgeneration reducing precision and thus the F1@M performance. However, F1@5 generally improves in beam search compared to greedy because only the top 5 keyphrases are considered in this metric. Either way, whether using greedy or beam search, in both One2One models and One2Set models, KPDROP enhances the base performance. Overall, for absent performance, KPDROP-R and KPDROP-A are both competitive against each other.

In Table 2, we show the recall performance for absent keyphrase generation when using high beam size. We find that applying any of the KPDROP techniques substantially improves the recall performance of absent keyphrase generation in all settings and datasets. In GRU One2Many and GRU One2One, KPDROP-R generally performs better than KPDROP-A. Interestingly, in Transformer One2Set KPDROP-A generally performs better than KPDROP-R across all datasets.

The above results validate our intuition that dropping keyphrases in KPDROP forces the models to deeply utilize the context information to learn to predict the dropped keyphrases, and thus, yields more robust models for absent keyphrase.

## 5.2 Results on Present Keyphrase Generation

In Table 3, we show the F1 performance for present keyphrase generation. As we hypothesized before, on greedy decoding KPDROP-R can significantly drop the performance of present keyphrases. However, KPDROP-A performs quite competitively against the baselines even in present keyphrase performance. Thus, KPDROP-A can be a balanced approach to boost absent performance without significantly downgrading the present performance. Interestingly, when combined with beam search, even KPDROP-R becomes competitive in present keyphrase generation. While, as we suggested before (in §2.1), KPDROP-R can create a tendency to undergenerate present keyphrases (given that many of them get turned artificially absent), beam search can fight against that tendency through overgeneration. This can sometimes lead to a “sweet spot" when beam search is combined with KPDROP-R making it competitive even in present performance for One2Many and One2One settings.

## 6 Semi-Supervised Experiments

We also investigate whether KPDROP has something to offer in a low-resource semi-supervised setting. We simulate a semi-supervised setting by randomly splitting the training set of KP20K into two parts. In one part, we keep 5000 samples with their original keyphrases intact, but in the other part, we keep the rest of the data after removing the original keyphrases. Thus, we are left with a low-resource (5000 samples) author-annotated training data, and a huge unlabelled corpus (rest

| Models | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 |
| **GRU One2Many** | | | | | | | | | | |
| Greedy | $\mathbf{27.4_{7}}$ | $\mathbf{27.1_{8}}$ | $38.1_{3}$ | $37.7_{6}$ | $\mathbf{34.7_{9}}$ | $\mathbf{34.2_{7}}$ | $\mathbf{29.6_{10}}$ | $\mathbf{29.1_{9}}$ | $\mathbf{37.3_{1}}$ | $\mathbf{37.0_{0}}$ |
| Greedy+KPD-R | $18.7_{3}$ | $18.7_{3}$ | $28.1_{3}$ | $28.1_{3}$ | $28.4_{6}$ | $28.4_{6}$ | $20.1_{14}$ | $20.1_{14}$ | $30.8_{2}$ | $30.8_{2}$ |
| Greedy+KPD-A | $25.1_{4}$ | $24.8_{3}$ | $\mathbf{38.5_{4}}$ | $\mathbf{38.2_{3}}$ | $34.1_{9}$ | $33.5_{8}$ | $27.9_{7}$ | $27.7_{6}$ | $37.0_{2}$ | $36.7_{2}$ |
| Beam | $38.7_{3}$ | $34.4_{6}$ | $37.0_{6}$ | $41.6_{5}$ | $25.6_{5}$ | $32.8_{3}$ | $31.9_{12}$ | $33.4_{3}$ | $32.2_{1}$ | $36.8_{0}$ |
| Beam+KPD-R | $34.9_{2}$ | $33.2_{2}$ | $\mathbf{42.0_{4}}$ | $41.0_{2}$ | $\mathbf{36.7_{3}}$ | $\mathbf{36.7_{2}}$ | $\mathbf{34.6_{9}}$ | $\mathbf{34.1_{9}}$ | $\mathbf{36.3_{1}}$ | $36.5_{1}$ |
| Beam+KPD-A | $\mathbf{38.9_{5}}$ | $\mathbf{34.6_{5}}$ | $39.4_{5}$ | $\mathbf{41.8_{7}}$ | $28.4_{0}$ | $34.2_{8}$ | $33.0_{17}$ | $33.2_{7}$ | $33.9_{2}$ | $\mathbf{37.1_{1}}$ |
| **GRU One2One** | | | | | | | | | | |
| Beam | $25.5_{1}$ | $\mathbf{30.6_{1}}$ | $16.5_{3}$ | $40.2_{10}$ | $12.4_{1}$ | $29.8_{1}$ | $16.5_{2}$ | $29.2_{16}$ | $13.1_{0}$ | $39.4_{1}$ |
| Beam+KPD-R | $\mathbf{29.6_{3}}$ | $30.4_{3}$ | $\mathbf{24.6_{4}}$ | $\mathbf{43.9_{3}}$ | $\mathbf{17.5_{1}}$ | $\mathbf{36.5_{2}}$ | $\mathbf{23.6_{4}}$ | $\mathbf{32.7_{10}}$ | $\mathbf{17.0_{2}}$ | $37.9_{0}$ |
| Beam+KPD-A | $26.3_{4}$ | $\mathbf{30.6_{2}}$ | $17.1_{0}$ | $38.9_{6}$ | $12.9_{0}$ | $29.4_{4}$ | $17.2_{5}$ | $30.0_{11}$ | $13.9_{1}$ | $\mathbf{39.6_{0}}$ |
| **Transformer One2Set** | | | | | | | | | | |
| Greedy | $\mathbf{32.2_{7}}$ | $\mathbf{31.3_{6}}$ | $43.7_{0}$ | $42.1_{3}$ | $35.2_{5}$ | $\mathbf{34.3_{11}}$ | $\mathbf{34.7_{2}}$ | $33.4_{7}$ | $39.2_{1}$ | $37.9_{4}$ |
| Greedy+KPD-R | $21.1_{3}$ | $21.0_{3}$ | $35.0_{9}$ | $34.9_{12}$ | $34.2_{7}$ | $34.2_{7}$ | $26.9_{7}$ | $27.0_{8}$ | $34.9_{2}$ | $34.8_{2}$ |
| Greedy+KPD-A | $30.6_{3}$ | $29.8_{4}$ | $\mathbf{44.4_{3}}$ | $\mathbf{42.6_{3}}$ | $\mathbf{35.3_{6}}$ | $34.0_{6}$ | $34.4_{5}$ | $\mathbf{33.6_{5}}$ | $\mathbf{39.6_{2}}$ | $\mathbf{38.5_{0}}$ |
| Beam | $21.7_{0}$ | $\mathbf{32.3_{4}}$ | $15.5_{1}$ | $\mathbf{42.3_{3}}$ | $11.0_{1}$ | $32.9_{6}$ | $16.3_{2}$ | $33.5_{8}$ | $10.9_{0}$ | $\mathbf{36.4_{5}}$ |
| Beam+KPD-R | $\mathbf{23.2_{2}}$ | $27.8_{8}$ | $\mathbf{21.0_{5}}$ | $40.0_{9}$ | $\mathbf{14.8_{8}}$ | $\mathbf{33.6_{7}}$ | $\mathbf{20.9_{5}}$ | $32.0_{5}$ | $\mathbf{15.0_{3}}$ | $35.1_{3}$ |
| Beam+KPD-A | $22.7_{1}$ | $32.2_{6}$ | $16.8_{2}$ | $41.8_{3}$ | $11.6_{2}$ | $32.3_{7}$ | $17.6_{4}$ | $\mathbf{34.3_{12}}$ | $11.7_{2}$ | $36.3_{0}$ |

Table 3: Present keyphrase performance (**F1**) for different models. KPD represents KPDrop. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$). We bold the best scores per block.

of the KP20K training set). Henceforth, we refer to the former low-resource labelled dataset as LR, and the latter unlabelled corpus as UC.

We investigate a pre-training based approach to utilize UC. Essentially, we pre-train our models first on UC and then fine-tune them on LR. For pre-training, similar to Ye and Wang (2018), we create synthetic labels using an unsupervised keyphrase extraction model. Unlike Ye and Wang (2018), we use a contemporary embedding-based keyphrase extraction model (Liang et al., 2021) to generate the synthetic keyphrases. Particularly, we rank the candidate keyphrases and keep the top 10. Note that in this pre-training setup, the synthetic keyphrases will only be present keyphrases because they are extracted from the input text. This is where KPDROP can make the unsupervised pre-training more interesting by creating artificial absent keyphrases by dropping of the synthetic present keyphrases (and simulating real data). We hypothesize that the application of KPDROP can help our models to learn more effective weights from UC in the self-supervised pre-training stage. We test the effectiveness of using KPDROP to augment self-supervised pre-training by testing the pre-trained model on the labelled test sets after fine-tuning on LR. During fine-tuning on LR, for all models, we always use KPDROP-A as we have already shown this to be beneficial in most supervised contexts. Hyperparameters are detailed in the Appendix A.1.

## 6.1 Semi-Supervised Results

For the semi-supervised experiments, report the beam search performance of GRU One2Many for the sake of brevity (greedy performance is in Appendix A.3). In Table 4, we show the absent performance of GRU One2Many in semi-supervised settings[3]. As we can see from the table, absent performance is near zero in almost all settings. Only when the model is pre-trained (PT) with KPDROP-A or KPDROP-R and then fine-tuned (FT) on LR (PT+KPD-R;FT or PT+KPD-A;FT), there is some degree of absent keyphrase generation. Thus, KPDROP is crucial for downstream absent keyphrase performance in this semi-supervised environment and for domains with low-resource annotated data.

In Table 4, we also show the present performance of GRU One2Many in semi-supervised settings. For present performance, we find that neither training only (FT) on LR nor training only (PT) on UC is as good as pre-training on UC and then fine-tuning the pre-trained model on LR (PT;FT). Although, one exception is the performance on Inspec which can be sometimes better when the model is zero-shotted after only training on UC. Either way, we again find that using KPDROP in the pre-training (PT+KPD-R;FT or PT+KPD-A;FT) setting signifi-

[3]Note that Liang et al. (2021) use an extraction model; thus it has no capabilities for absent keyphrase generation

| | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **F@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** |
| **Absent Keyphrase Performance** | | | | | | | | | | |
| Unsupervised Extraction Model (Liang et al., 2021) | | | | | | | | | | |
| | — | 0.0 | — | 0.0 | — | 0.0 | — | 0.0 | — | 0.0 |
| GRU One2Many Models (Beam Search) | | | | | | | | | | |
| PT | $0.0_1$ | $0.0_1$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_1$ | $0.0_1$ | $0.0_0$ | $0.0_0$ | $0.0_0$ |
| PT+KPD-R | $0.7_1$ | $0.7_1$ | $0.8_2$ | $0.7_1$ | $0.6_2$ | $0.5_2$ | $0.3_1$ | $0.2_2$ | $0.8_1$ | $0.8_1$ |
| PT+KPD-A | $0.1_0$ | $0.1_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ |
| FT | $0.7_1$ | $0.6_1$ | $1.1_0$ | $1.1_0$ | $0.5_0$ | $0.6_0$ | $0.3_1$ | $0.3_1$ | $0.7_1$ | $0.7_1$ |
| PT; FT | $0.7_2$ | $0.7_2$ | $0.5_4$ | $0.5_4$ | $0.1_0$ | $0.1_0$ | $0.1_1$ | $0.1_1$ | $0.4_1$ | $0.3_1$ |
| PT+KPD-A; FT | $0.9_2$ | $0.9_2$ | $2.5_3$ | $2.6_4$ | $1.8_2$ | $1.8_2$ | $1.8_1$ | $1.8_1$ | $1.8_1$ | $1.8_1$ |
| PT+KPD-R; FT | $\mathbf{1.7_3}$ | $\mathbf{1.7_3}$ | $\mathbf{2.9_3}$ | $\mathbf{2.8_3}$ | $\mathbf{2.8_6}$ | $\mathbf{2.8_5}$ | $\mathbf{2.0_1}$ | $\mathbf{2.0_1}$ | $\mathbf{2.7_2}$ | $\mathbf{2.7_2}$ |
| **Present Keyphrase Performance** | | | | | | | | | | |
| Unsupervised Extraction Model (Liang et al., 2021) | | | | | | | | | | |
| | — | 32.6 | — | 20.8 | — | 18.1 | — | 13.02 | — | 17.9 |
| GRU One2Many Models (Beam Search) | | | | | | | | | | |
| PT | $\mathbf{40.9_3}$ | $36.0_3$ | $20.6_2$ | $22.4_{10}$ | $18.5_2$ | $17.9_2$ | $22.8_5$ | $22.2_{11}$ | $16.0_1$ | $17.9_2$ |
| PT+KPD-R | $36.2_{13}$ | $34.3_4$ | $24.8_{13}$ | $23.6_{20}$ | $19.2_1$ | $18.9_4$ | $25.1_6$ | $23.2_{11}$ | $18.6_8$ | $18.0_7$ |
| PT+KPD-A | $40.2_3$ | $35.7_4$ | $21.0_6$ | $22.8_5$ | $18.8_5$ | $19.8_2$ | $23.4_6$ | $22.7_6$ | $16.1_2$ | $17.9_2$ |
| FT | $27.8_{11}$ | $26.5_9$ | $32.6_2$ | $31.8_3$ | $26.6_8$ | $26.4_8$ | $27.8_8$ | $26.7_7$ | $27.1_4$ | $26.9_4$ |
| PT; FT | $36.8_{17}$ | $33.5_{25}$ | $33.8_3$ | $33.5_2$ | $26.2_{12}$ | $27.5_6$ | $29.1_5$ | $27.6_7$ | $26.2_3$ | $27.2_2$ |
| PT+KPD-A; FT | $39.9_5$ | $36.0_{10}$ | $\mathbf{36.3_4}$ | $\mathbf{35.7_2}$ | $29.2_4$ | $29.6_3$ | $\mathbf{32.0_4}$ | $\mathbf{30.8_{12}}$ | $28.2_3$ | $28.8_2$ |
| PT+KPD-R; FT | $40.0_{11}$ | $\mathbf{36.9_{10}}$ | $36.2_{11}$ | $34.6_9$ | $\mathbf{31.5_3}$ | $\mathbf{31.1_7}$ | $30.9_5$ | $29.9_{14}$ | $\mathbf{29.7_7}$ | $\mathbf{29.8_8}$ |

Table 4: Absent and present keyphrase performance using Beam Search for GRU One2Many models in a semi-supervised setup. KPD represents KPDrop. PT represents pre-training on the synthetic data (UC). PT+KPD-R or PT+KPD-A represents pre-training on UC with KPD-R or KPD-A repsectively. FT represents fine tuning or training on the low resource labelled data (LR). PT (or PT+KPD-A or PT+KPD-R) followed by "; FT" represents that the pre-training was followed by fine-tuning of the pre-trained model on LR. We bold the overall best scores. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$).

cantly boosts present performance after fine-tuning.

Thus, using KPDROP to make the pre-training stage more challenging by pushing the model to predict missing present keyphrases helps not only with absent keyphrase performance but also with present keyphrase performance after fine-tuning. In between KPDROP-R and KPDROP-A, the former generally performs better in the pre-training stage. Thus, during pre-training it is better to *replace* synthetic labels of fully present keyphrases with its KPDropped version. In Appendix A.3, we also observe similar patterns from other models (One2One and One2Set) in semi-supervised settings.

## 7 Preliminary Experiments on Pre-trained Models

We also did a few preliminary experiments on applying KPDROP-R to a large pre-trained Seq2Seq language model, in particular, T5 (Raffel et al., 2020). We present the results in Appendix A.4. Consistent with previous results, we find that KPDROP-R increases absent performance for T5 as well. However, overall, we found the performance of T5 baseline to be limited compared to trained-from-sratch models like Transformer One2Set. Similarly, other reported performances on pre-trained models have been generally lower than Transformers One2Set too. For example, even after large scale specialized pre-training for keyphrases, Kulkarni et al. (2022) reports only comparable performance on present keyphrases to Transformer One2Set (Ye et al., 2021b) and much less in absent performance. On the other hand, using pre-trained models, Wu et al. (2021)[4] achieve comparable on absent performance with One2Set under greedy search, but much less in present performance. However, it should not be too difficult to adapt a pre-trained model into a one2set framework during fine-tuning. This can be a promising future direction and form a stronger base model for KPDROP.

## 8 Related Work

There is a wide variety of approaches (Hasan and Ng, 2014; Caragea et al., 2014; Das Gollapalli and

[4]We are referring to the latest ArXiv version (v2) which holds the globally latest version of the paper.

Caragea, 2014; Gollapalli and Li, 2016; Sterckx et al., 2016; Florescu and Caragea, 2017; Zhang et al., 2017; Boudin, 2018; Mahata et al., 2018; Sun et al., 2019; Al-Zaidy et al., 2019; Campos et al., 2020; Santosh et al., 2020; Sahrawat et al., 2020; Sun et al., 2020; Song et al., 2021; Patel and Caragea, 2021) for keyphrase extraction exclusively. Meng et al. (2017) diverged from pure extractive methods by introducing a seq2seq model (CopyRNN) for generation of both present and absent keyphrases. Chen et al. (2018) extended CopyRNN with keyphrase correlation constraints and Zhao and Zhang (2019) extended it with linguistic constraints. Ye and Wang (2018); Wu et al. (2022) investigated keyphrase generation (KG) in semi-supervised or resource-constrained settings. Chen et al. (2019b) used a title-guided encoding method for better KG. Wang et al. (2019) incorporated a topic-model to enhance KG. Yuan et al. (2020) extended CopyRNN by introducing the CatSeq model that can generate a concatenation of dynamically determined variable number of keyphrases. Chan et al. (2019); Luo et al. (2021) improved KG using reinforcement learning whereas Swaminathan et al. (2020a,b); Lancioni et al. (2020) do so using GANs. A few approaches (Chen et al., 2019a; Diao et al., 2020; Kim et al., 2021; Ye et al., 2021a) augmented KG with information from retrieved documents.

Multiple approaches (Chen et al., 2019a; Liu et al., 2020; Ahmad et al., 2021; Zhao et al., 2021; Wu et al., 2021) took a joint-training or multi-tasking approach to do both present keyphrase extraction and absent keyphrase generation. Chen et al. (2020) and Ye et al. (2021b) changed the decoder to better respect the structure of keyphrases. Luo et al. (2020) changed the encoder to better respect the input document structure. Huang et al. (2021) proposed a new beam-search-based adaptive decoding method. Meng et al. (2021); Kulkarni et al. (2022) investigated pre-training objectives for KG. Both, however, rely on labelled pre-training data.

## 9 Conclusion

Our proposal, KPDrop, randomly drops present keyphrases from a document to turn them artificially absent. This encourages the model to learn to better exploit the context in the input to be able to infer keyphrases that are absent from the text but otherwise topically relevant. The results show that KPDrop serves as a simple model-agnostic method to substantially improve absent (and sometimes, present) keyphrase performance in both supervised and semi-supervised (low resource) settings when large annotated datasets for keyphrase generation are not available. In future, we would like to explore integration of KPDrop with large-scale pre-training.

## 10 Limitations

KPDrop is a simple yet effective approach to improving performance of keyphrase generation in both large datasets and low resource datasets, which makes it applicable to a wide range of domains where keyphrases are necessary. However, one limitation of KPDrop (especially KPDrop-A) is that it can increase the computation cost during training because both the effective mini-batch size and the effective training dataset size per epoch is doubled through data augmentation. Yet, most data augmentation techniques share the same limitation.

In addition, KPDrop-R can potentially harm the performance of present keyphrases in some contexts (especially when using greedy search in a supervised setting). To address this, we can simply use the absent predictions of the model trained with KPDrop and the present predictions of the model trained without KPDrop.

## 11 Ethics Statement

Our technique is specifically designed to improve keyphrase generation. Keyphrase generation is a well-established traditional NLP task that is useful in several application contexts related to organization of information. We do not foresee any immediate ethical concern following from our contribution in this area.

## Acknowledgements

This research is supported in part by NSF CAREER award #1802358, NSF CRI award #1823292, NSF IIS award #2107518, and UIC Discovery Partners Institute (DPI) award. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF or DPI. We thank AWS for computational resources used for this study. We also thank our anonymous reviewers for their constructive feedback and suggestions.

## 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.

Rabah Al-Zaidy, Cornelia Caragea, and C. Lee Giles. 2019. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *Proceedings of The Web Conference (WWW 2019), San Francisco, California, USA*.

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. 2019. Memory efficient adaptive optimization. *Advances in Neural Information Processing Systems*, 32:9749–9758.

Haoli Bai, Zhuangbin Chen, Michael R. Lyu, Irwin King, and Zenglin Xu. 2018. Neural relational topic models for scientific article analysis. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 27–36, New York, NY, USA. Association for Computing Machinery.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.

Florian Boudin and Ygor Gallina. 2021. Redefining absent keyphrases and their effect on retrieval effectiveness. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4185–4193, Online. 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, Online. Association for Computational Linguistics.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1435–1446, Doha, Qatar. Association for 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.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Shizhe Diao, Yan Song, and Tong Zhang. 2020. Keyphrase generation with cross-document attention. *ArXiv*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

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

George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971.

Sujatha Das Gollapalli and Xiaoli Li. 2016. Keyphrase extraction using sequential labeling. *ArXiv*, abs/1608.00329.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273, Baltimore, Maryland. Association for Computational Linguistics.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep networks with stochastic depth. In *Computer Vision – ECCV 2016*, pages 646–661, Cham. Springer International Publishing.

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.

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.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

James M. Kates and Kathryn H. Arehart. 2014. The hearing-aid speech perception index (haspi). *Speech Communication*, 65:75–93.

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.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction. *University of Trento*.

H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

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.

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.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Yichao Luo, Zhengyan Li, Bingning Wang, Xiaoyu Xing, Qi Zhang, and Xuanjing Huang. 2020. Sensenet: Neural keyphrase generation with document structure. *ArXiv*, abs/2012.06754.

Yichao Luo, Yige Xu, Jiacheng Ye, Xipeng Qiu, and Qi Zhang. 2021. Keyphrase generation with fine-grained evaluation-guided reinforcement learning. In *Proceedings of EMNLP findings*.

Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. 2018. Key2Vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 634–639, New Orleans, Louisiana. Association for Computational Linguistics.

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.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Sujian Li, and Houfeng Wang. 2012. Entity-centric topic-oriented opinion summarization in twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 379–387.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pages 317–326, Berlin, Heidelberg. Springer Berlin Heidelberg.

Krutarth Patel and Cornelia Caragea. 2021. Exploiting position and contextual word embeddings for keyphrase extraction from scientific papers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1585–1591, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar Singla, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. *Information Retrieval, 42nd European Conference on IR Research, ECIR 2020,*, 12036:328 – 335.

T.y.s.s Santosh, Debarshi Kumar Sanyal, Plaban Kumar Bhowmick, and Partha Pratim Das. 2020. SaSAKE: Syntax and semantics aware keyphrase extraction from research papers. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5372–5383, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mingyang Song, Liping Jing, and Lin Xiao. 2021. Importance Estimation from Multiple Perspectives for Keyphrase Extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2726–2736, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

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

Si Sun, Zhenghao Liu, Chenyan Xiong, Zhiyuan Liu, and Jie Bao. 2020. Joint keyphrase chunking and salience ranking with bert. *ArXiv*.

Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 755–764, New York, NY, USA. Association for Computing Machinery.

Avinash Swaminathan, Raj Kuwar Gupta, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, and Rajiv Ratn Shah. 2020a. Keyphrase generation for scientific articles using gans (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(10):13931–13932.

Avinash Swaminathan, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, Rajiv Ratn Shah, and Amanda Stent. 2020b. 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.

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.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Di Wu, Wasi Uddin Ahmad, Sunipa Dev, and Kai-Wei Chang. 2022. 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.

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.

Yuxiang Zhang, Yaocheng Chang, Xiaoqing Liu, Sujatha Das Gollapalli, Xiaoli Li, and Chunjing Xiao. 2017. Mike: Keyphrase extraction by integrating multidimensional information. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1349–1358, New York, NY, USA. Association for Computing Machinery.

Jing Zhao, Junwei Bao, Yifan Wang, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021. SGG: Learning to select, guide, and generate for keyphrase generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5717–5726, Online. 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.

Özlem Ünlü and Aydın Çetin. 2019. A survey on keyword and key phrase extraction with deep learning. In *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–6.

## A Appendix

### A.1 Hyperparameters

For GRU One2Many (CatSeq) we use hyperparameters similar to prior works (Chan et al., 2019). Like prior works (Yuan et al., 2020; Meng et al., 2021), when using beam search on one2many models we use a beam size of 50. We use the same model hyperparameters of GRU One2One models as used for GRU One2Many. However, during beam search we use a beam size of 200 as is standard for one2one models in prior work (Meng et al., 2017, 2021). A lower beam size (50 instead of 200) is used for one2many models because they can generate multiple keyphrases per beam. So a lower size is used to make them more comparable with one2one models with higher beam size. For Transformer One2Set model we use the same hyperparameters as Ye et al. (2021b). Given the similarity of One2Set models with One2Many models in their ability to generate multiple keyphrases per beam, we also use a beam size of 50 for for Transformer One2Set. Anytime we use beam search, we also use length normalization on beam search with a length co-efficient of $0.8$. We use the same settings during semi-supervised pre-training or fine-tuning. We tested KPDROP-R rates among $\{0.3, 0.5, 0.7, 0.9, 1.0\}$ on GRU One2Many over the validation set after one epoch training on the full KP20K training set. We found $0.7$ to be the best performing rate for validation absent performance. We use this same rate for KPDROP-A and other models in both supervised and semi-supervised settings. All the experiments were done in a single Nvidia RTX A6000.

### A.2 More Evaluations

Chan et al. (2019) modified the original $F_1@5$ (as used in the main paper and other prior works (Meng et al., 2017; Yuan et al., 2020)) such that the denominator in the precision is always set to 5 even when the total predictions are less than 5. To avoid confusion and better distinguish from the original $F_1@5$ we refer to the modified metric as $F_1@5C$. Throughout the paper, for the sake of brevity we mainly report performance in $F_1@5$ instead of $F_1@5C$. This is because although $F_1@5C$ achieve the goal of differentiating itself from $F_1@M$ reports in greedy search contexts, it can be a little misleading. For example, $F_1@5C$ can artificially penalize the model for predicting less than 5 keyphrases even when the ground truth itself contains less than 5 keyphrases. Nevertheless, in Table 5, we still report the $F_1@5C$ present and absent performance of our models from our supervised experiments for the sake of better comparison with prior works that use $F_1@5C$.

### A.3 More Semi-Supervised Experiments

In Table 6 we present the greedy decoding performance of GRU One2Many models in semi-supervised settings. In Table 7 we present the performance of GRU One2One models (beam search) in semi-supervised settings. In Table 8 and 9 we present the greedy decoding performance and beam decoding performance of Transformer One2Set mdoels in semi-supervised settings respectively. Gnerally, we notice similar patterns across all the models as were found and discussed for GRU One2Many models in §6.1. Overall, KPDROP-R consistently serves as a crucial ingredient in the pre-training stage to enable substantially improved downstream performance in both present and absent keyphrase generation.

### A.4 Preliminary Experiments on Additional Baselines

We also present the results of applying KPDROP-R on a large pre-trained model T5 (Raffel et al., 2020), and another specialized KG model, ExHiRD (Chen et al., 2020) in Table 10. In both cases, we see the promise of KPDROP-R in improving the absent performance.

Below we describe the hyperparameters that were used for the preliminary experiments described in this section.

For ExHiRD, we use the same hyperparameters as in the original paper (Chen et al., 2020)[5]. For T5, we use a maximum of 10 epochs, early stopping with a patience of 2 (the training is terminated if validation loss does not improve for 2 consecutive epochs), a batch size as 64, a maximum gradient norm clipping af 5, and SM3 (Anil et al., 2019) as the optimizer. The initial learning rate for T5 was set to be $0.1$. We apply learning rate (lr) warmup as follows [6]:

$$lr_s = lr_0 \cdot min(1, (s/w)^2) \tag{1}$$

$lr_s$ indicates the learning rate at step $s$. $lr_0$ is the initial learning rate $(0.1)$. $s$ indicates the current

[5]https://github.com/Chen-Wang-CUHK/ExHiRD-DKG

[6]based on the recommended procedure for SM3 (https://github.com/google-research/google-research/tree/master/sm3).

| Models | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F1@5C | | F1@5C | | F1@5C | | F1@5C | | F1@5C | |
| | Pre | Abs | Pre | Abs | Pre | Abs | Pre | Abs | Pre | Abs |
| **GRU One2Many** | | | | | | | | | | |
| Greedy | $\mathbf{22.1_{6}}$ | $0.7_{1}$ | $\mathbf{31.2_{3}}$ | $1.6_{3}$ | $\mathbf{26.4_{5}}$ | $1.8_{3}$ | $\mathbf{24.6_{8}}$ | $1.6_{2}$ | $\mathbf{29.3_{0}}$ | $1.6_{0}$ |
| Greedy+KPD-R | $12.5_{3}$ | $0.6_{0}$ | $18.4_{1}$ | $1.7_{3}$ | $15.6_{3}$ | $\mathbf{2.6_{1}}$ | $13.9_{12}$ | $\mathbf{1.8_{1}}$ | $17.7_{1}$ | $1.9_{0}$ |
| Greedy+KPD-A | $19.7_{4}$ | $\mathbf{0.8_{1}}$ | $30.6_{6}$ | $\mathbf{1.8_{3}}$ | $24.7_{2}$ | $2.4_{1}$ | $23.2_{6}$ | $\mathbf{1.8_{3}}$ | $28.1_{5}$ | $\mathbf{2.0_{2}}$ |
| Beam | $\mathbf{34.1_{5}}$ | $2.3_{0}$ | $41.3_{6}$ | $4.0_{2}$ | $32.6_{3}$ | $5.2_{2}$ | $\mathbf{33.3_{3}}$ | $3.2_{1}$ | $\mathbf{36.6_{0}}$ | $4.4_{1}$ |
| Beam+KPD-R | $30.3_{2}$ | $\mathbf{2.6_{2}}$ | $37.8_{3}$ | $\mathbf{5.7_{1}}$ | $31.9_{2}$ | $\mathbf{6.6_{2}}$ | $32.4_{13}$ | $\mathbf{4.3_{4}}$ | $32.3_{1}$ | $\mathbf{5.4_{0}}$ |
| Beam+KPD-A | $34.0_{5}$ | $2.5_{1}$ | $\mathbf{41.6_{7}}$ | $5.0_{6}$ | $\mathbf{33.8_{9}}$ | $5.5_{2}$ | $33.1_{7}$ | $\mathbf{4.3_{3}}$ | $\mathbf{36.6_{0}}$ | $5.0_{0}$ |
| **GRU One2One** | | | | | | | | | | |
| Beam | $30.5_{2}$ | $2.8_{1}$ | $40.2_{10}$ | $5.7_{7}$ | $29.8_{1}$ | $5.9_{3}$ | $29.2_{16}$ | $3.8_{1}$ | $39.4_{1}$ | $6.2_{0}$ |
| Beam+KPD-R | $30.4_{3}$ | $\mathbf{2.9_{3}}$ | $\mathbf{43.9_{3}}$ | $\mathbf{7.5_{3}}$ | $\mathbf{36.5_{2}}$ | $\mathbf{7.8_{2}}$ | $\mathbf{32.7_{10}}$ | $\mathbf{4.9_{5}}$ | $37.9_{0}$ | $6.5_{1}$ |
| Beam+KPD-A | $\mathbf{30.6_{2}}$ | $2.7_{1}$ | $38.9_{6}$ | $6.7_{2}$ | $29.4_{4}$ | $6.5_{3}$ | $30.0_{11}$ | $4.1_{3}$ | $\mathbf{39.6_{0}}$ | $\mathbf{6.8_{0}}$ |
| **Transformer One2Set** | | | | | | | | | | |
| Greedy | $\mathbf{27.6_{5}}$ | $1.9_{3}$ | $\mathbf{38.8_{4}}$ | $4.3_{5}$ | $\mathbf{30.9_{9}}$ | $4.1_{2}$ | $\mathbf{31.2_{1}}$ | $2.6_{2}$ | $\mathbf{34.4_{3}}$ | $3.3_{0}$ |
| Greedy+KPD-R | $15.3_{2}$ | $2.0_{0}$ | $26.0_{9}$ | $\mathbf{5.3_{5}}$ | $22.2_{4}$ | $\mathbf{5.9_{5}}$ | $20.7_{7}$ | $\mathbf{3.9_{3}}$ | $24.0_{2}$ | $\mathbf{4.5_{1}}$ |
| Greedy+KPD-A | $25.7_{3}$ | $\mathbf{2.1_{1}}$ | $38.0_{2}$ | $5.2_{8}$ | $29.5_{7}$ | $4.6_{4}$ | $30.3_{7}$ | $3.6_{1}$ | $33.9_{3}$ | $4.2_{1}$ |
| Beam | $\mathbf{32.3_{4}}$ | $3.3_{2}$ | $\mathbf{42.3_{3}}$ | $7.0_{5}$ | $32.9_{6}$ | $6.7_{5}$ | $33.5_{8}$ | $4.7_{4}$ | $\mathbf{36.4_{5}}$ | $5.8_{0}$ |
| Beam+KPD-R | $27.8_{8}$ | $2.4_{1}$ | $40.0_{9}$ | $7.2_{5}$ | $\mathbf{33.7_{7}}$ | $7.3_{2}$ | $32.0_{5}$ | $5.2_{4}$ | $35.1_{3}$ | $6.1_{0}$ |
| Beam+KPD-A | $32.2_{6}$ | $\mathbf{3.6_{2}}$ | $41.8_{3}$ | $\mathbf{7.9_{4}}$ | $32.3_{7}$ | $\mathbf{7.8_{2}}$ | $\mathbf{34.3_{12}}$ | $\mathbf{5.3_{4}}$ | $36.3_{0}$ | $\mathbf{6.7_{0}}$ |

Table 5: Present and absent keyphrase performance for different models. Pre represents present performance and Abs represents absent performance. KPD represents KPDrop. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$). We bold the best scores per block.

update step number. $w$ indicates total warmup steps (set as 2000). The initial learning rate (0.1) was tuned using grid search based on validation loss among the following choices: $\{0.1, 0.01, 0.001\}$. For each trial during hyperparameter optimization we use a maximum of 1 epoch. We only tune the baselines (where KPDROP is unapplied). We do not separately tune other hyperparameters when KPDROP is applied.

We use the T5 implementation as provided by (Wolf et al., 2020). Both ExHiRD and T5 models are trained using teacher forcing mechanism. During inference, we set the maximum phrase length as 50 for both. Keyphrase tokens are greedily generated during inference. A KPDROP rate of 0.7 was used - same as the other previous experiments.

| | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| Models | F@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 |
| **GRU One2Many Models (Greedy Search)** | | | | | | | | | | |
| **Absent Keyphrase Performance** | | | | | | | | | | |
| PT | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ |
| PT+KPD-R | $0.2_1$ | $0.2_1$ | $0.2_2$ | $0.2_2$ | $0.1_0$ | $0.1_0$ | $0.0_0$ | $0.0_0$ | $0.3_1$ | $0.3_1$ |
| PT+KPD-A | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ |
| FT | $0.1_1$ | $0.1_1$ | $0.0_0$ | $0.1_0$ | $0.2_0$ | $0.2_0$ | $0.3_0$ | $0.3_0$ | $0.1_0$ | $0.1_0$ |
| PT; FT | $0.0_1$ | $0.0_1$ | $0.2_1$ | $0.2_1$ | $0.0_0$ | $0.0_0$ | $0.1_1$ | $0.1_1$ | $0.1_0$ | $0.1_0$ |
| PT+KPD-A; FT | $0.2_1$ | $0.2_1$ | $\mathbf{0.8_4}$ | $\mathbf{0.8_4}$ | $0.4_1$ | $0.4_1$ | $\mathbf{0.7_0}$ | $\mathbf{0.7_0}$ | $0.6_0$ | $0.6_0$ |
| PT+KPD-R; FT | $\mathbf{0.8_3}$ | $\mathbf{0.8_3}$ | $0.5_2$ | $0.5_2$ | $\mathbf{0.5_0}$ | $\mathbf{0.5_0}$ | $0.4_2$ | $0.4_2$ | $\mathbf{0.8_1}$ | $\mathbf{0.8_1}$ |
| **Present Keyphrase Performance** | | | | | | | | | | |
| PT | $35.0_7$ | $34.2_6$ | $23.7_2$ | $23.7_2$ | $20.1_4$ | $20.3_2$ | $24.5_6$ | $\mathbf{24.8_5}$ | $18.4_1$ | $18.7_1$ |
| PT+KPD-R | $32.0_4$ | $32.0_5$ | $25.3_3$ | $24.8_5$ | $20.5_5$ | $20.3_3$ | $\mathbf{25.0_{11}}$ | $24.7_{11}$ | $19.3_2$ | $19.0_3$ |
| PT+KPD-A | $\mathbf{35.5_5}$ | $\mathbf{34.4_4}$ | $23.8_9$ | $23.9_7$ | $19.3_6$ | $20.0_3$ | $24.5_{15}$ | $24.3_{10}$ | $18.2_4$ | $18.5_3$ |
| FT | $14.3_{12}$ | $14.3_{12}$ | $24.7_{21}$ | $24.7_{21}$ | $23.2_{12}$ | $23.2_{12}$ | $15.8_2$ | $15.8_2$ | $22.8_8$ | $22.8_8$ |
| PT; FT | $20.9_{18}$ | $20.9_{18}$ | $27.2_{16}$ | $27.2_{16}$ | $25.5_{10}$ | $25.5_{10}$ | $21.0_5$ | $21.0_5$ | $25.1_7$ | $25.1_7$ |
| PT+KPD-A; FT | $22.0_3$ | $22.0_3$ | $\mathbf{30.1_7}$ | $\mathbf{30.1_7}$ | $26.5_7$ | $26.5_7$ | $23.0_6$ | $23.0_6$ | $27.0_2$ | $27.0_2$ |
| PT+KPD-R; FT | $20.5_4$ | $20.5_4$ | $\mathbf{30.1_8}$ | $\mathbf{30.1_8}$ | $\mathbf{27.8_4}$ | $\mathbf{27.8_4}$ | $22.3_{12}$ | $22.3_{12}$ | $\mathbf{27.1_2}$ | $\mathbf{27.1_2}$ |

Table 6: Absent and present keyphrase performance using Greedy Search for GRU One2Many models in a semi-supervised setup. KPD represents KPDrop. PT represents pre-training on the synthetic data (UC). PT+KPD-R or PT+KPD-A represents pre-training on UC with KPD-R or KPD-A repsectively. FT represents fine tuning or training on the low resource labelled data (LR). PT (or PT+KPD-A or PT+KPD-R) followed by "; FT" represents that the pre-training was followed by fine-tuning of the pre-trained model on LR. We bold the overall best scores. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$).

| | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| Models | F@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 |
| **GRU One2One (Beam Search)** | | | | | | | | | | |
| **Absent Keyphrase Performance** | | | | | | | | | | |
| PT | $0.0_0$ | $0.2_1$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.0_0$ | $0.1_1$ | $0.0_0$ | $0.1_0$ |
| PT+KPD-R | $0.4_0$ | $1.2_1$ | $0.7_0$ | $0.9_2$ | $0.6_0$ | $1.6_2$ | $0.9_1$ | $0.9_2$ | $0.4_0$ | $1.3_2$ |
| PT+KPD-A | $0.1_0$ | $0.2_1$ | $0.1_0$ | $0.1_1$ | $0.0_0$ | $0.0_0$ | $0.1_0$ | $0.0_0$ | $0.1_0$ | $0.1_0$ |
| FT | $0.2_0$ | $0.3_0$ | $0.6_0$ | $1.2_2$ | $0.3_0$ | $0.9_1$ | $0.4_0$ | $0.4_0$ | $0.3_0$ | $0.8_1$ |
| PT; FT | $0.2_0$ | $0.6_1$ | $0.6_1$ | $1.3_3$ | $0.4_0$ | $1.0_1$ | $0.3_1$ | $0.5_3$ | $0.3_0$ | $0.8_1$ |
| PT+KPD-A; FT | $0.3_0$ | $1.0_1$ | $0.7_0$ | $1.4_2$ | $0.5_0$ | $1.4_1$ | $0.6_1$ | $1.1_3$ | $0.4_0$ | $1.3_0$ |
| PT+KPD-R; FT | $\mathbf{0.6_0}$ | $\mathbf{1.9_2}$ | $\mathbf{1.4_0}$ | $\mathbf{4.7_2}$ | $\mathbf{1.1_0}$ | $\mathbf{4.4_2}$ | $\mathbf{1.2_0}$ | $\mathbf{3.0_2}$ | $\mathbf{0.8_0}$ | $\mathbf{3.7_0}$ |
| **Present Keyphrase Performance** | | | | | | | | | | |
| PT | $14.9_1$ | $34.1_8$ | $9.3_0$ | $25.1_4$ | $6.7_0$ | $20.3_6$ | $11.0_1$ | $25.2_3$ | $6.6_1$ | $19.0_1$ |
| PT+KPD-R | $\mathbf{21.6_6}$ | $33.6_5$ | $12.6_3$ | $25.3_6$ | $9.8_1$ | $19.8_2$ | $14.6_7$ | $23.1_8$ | $9.2_1$ | $18.4_2$ |
| PT+KPD-A | $15.7_2$ | $\mathbf{34.5_5}$ | $9.7_1$ | $25.9_7$ | $7.0_1$ | $20.3_5$ | $11.1_2$ | $25.4_{11}$ | $6.8_1$ | $19.0_5$ |
| FT | $20.6_{13}$ | $20.5_{22}$ | $\mathbf{17.8_{15}}$ | $31.9_{20}$ | $\mathbf{12.5_9}$ | $23.4_{17}$ | $\mathbf{16.2_{18}}$ | $21.8_{24}$ | $\mathbf{13.5_8}$ | $25.2_8$ |
| PT; FT | $20.6_3$ | $26.1_7$ | $13.7_5$ | $36.6_5$ | $10.1_1$ | $28.2_6$ | $13.9_3$ | $28.1_{15}$ | $10.0_2$ | $29.5_1$ |
| PT+KPD-A; FT | $20.6_{10}$ | $27.8_{10}$ | $13.8_8$ | $35.2_{17}$ | $10.1_6$ | $28.9_9$ | $14.0_7$ | $28.7_8$ | $9.9_6$ | $29.2_4$ |
| PT+KPD-R; FT | $21.4_3$ | $29.4_8$ | $14.3_2$ | $\mathbf{39.0_8}$ | $10.5_2$ | $\mathbf{30.7_4}$ | $15.0_1$ | $\mathbf{30.4_4}$ | $10.4_2$ | $\mathbf{32.5_1}$ |

Table 7: Absent and present keyphrase performance using Beam Search for GRU One2One models in a semi-supervised setup. KPD represents KPDrop. PT represents pre-training on the synthetic data (UC). PT+KPD-R or PT+KPD-A represents pre-training on UC with KPD-R or KPD-A repsectively. FT represents fine tuning or training on the low resource labelled data (LR). PT (or PT+KPD-A or PT+KPD-R) followed by "; FT" represents that the pre-training was followed by fine-tuning of the pre-trained model on LR. We bold the overall best scores. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$).

| | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **F@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** |
| **Transformer One2Set (Greedy Search)** | | | | | | | | | | |
| **Absent Keyphrase Performance** | | | | | | | | | | |
| PT | $0.2_1$ | $0.2_1$ | $0.1_1$ | $0.1_1$ | $0.3_1$ | $0.3_1$ | $0.0_0$ | $0.0_0$ | $0.1_0$ | $0.1_0$ |
| PT+KPD-R | $0.6_0$ | $0.6_0$ | $0.8_1$ | $0.8_1$ | $0.8_5$ | $0.7_5$ | $0.3_2$ | $0.3_2$ | $0.3_0$ | $0.6_0$ |
| PT+KPD-A | $0.3_1$ | $0.3_1$ | $0.2_1$ | $0.2_1$ | $0.2_0$ | $0.2_0$ | $0.1_1$ | $0.1_1$ | $0.2_0$ | $0.2_0$ |
| FT | $0.2_3$ | $0.2_3$ | $0.4_3$ | $0.4_3$ | $0.7_3$ | $0.7_3$ | $0.3_1$ | $0.3_1$ | $0.5_1$ | $0.5_1$ |
| PT; FT | $0.3_2$ | $0.3_2$ | $0.5_1$ | $0.5_1$ | $0.4_2$ | $0.4_2$ | $0.0_0$ | $0.0_0$ | $0.4_2$ | $0.4_2$ |
| PT+KPD-A; FT | $0.5_2$ | $0.5_2$ | $0.7_2$ | $0.7_2$ | $1.1_1$ | $1.1_1$ | $1.1_5$ | $1.1_5$ | $1.2_0$ | $1.2_0$ |
| PT+KPD-R; FT | $\mathbf{1.1_1}$ | $\mathbf{1.1_1}$ | $\mathbf{2.9_9}$ | $\mathbf{2.9_9}$ | $\mathbf{2.5_5}$ | $\mathbf{2.5_5}$ | $\mathbf{2.0_6}$ | $\mathbf{2.0_6}$ | $\mathbf{2.4_1}$ | $\mathbf{2.4_1}$ |
| **Present Keyphrase Performance** | | | | | | | | | | |
| PT | $34.6_5$ | $28.6_{17}$ | $25.9_3$ | $23.3_{10}$ | $20.1_4$ | $17.8_4$ | $27.1_4$ | $23.3_7$ | $18.9_1$ | $17.2_4$ |
| PT+KPD-R | $\mathbf{35.7_6}$ | $\mathbf{31.7_{13}}$ | $26.3_9$ | $23.9_{11}$ | $20.4_{10}$ | $18.3_5$ | $26.0_7$ | $23.0_{18}$ | $19.3_8$ | $17.8_7$ |
| PT+KPD-A | $34.7_9$ | $28.9_5$ | $25.7_4$ | $24.1_{12}$ | $19.8_2$ | $18.5_{10}$ | $26.8_4$ | $23.5_{10}$ | $19.2_1$ | $18.0_{10}$ |
| FT | $4.6_7$ | $4.6_7$ | $11.5_{41}$ | $11.5_{41}$ | $9.5_{11}$ | $9.5_{11}$ | $6.1_{17}$ | $6.1_{17}$ | $8.5_{24}$ | $8.5_{24}$ |
| PT; FT | $20.6_{14}$ | $20.6_{14}$ | $29.2_{20}$ | $29.2_{20}$ | $24.9_{11}$ | $24.9_{11}$ | $23.2_{16}$ | $23.2_{16}$ | $24.7_{15}$ | $24.7_{15}$ |
| PT+KPD-A; FT | $23.3_5$ | $23.3_5$ | $32.5_{10}$ | $32.3_9$ | $27.1_9$ | $27.1_8$ | $26.7_{11}$ | $26.7_{12}$ | $27.5_2$ | $27.4_2$ |
| PT+KPD-R; FT | $26.4_{19}$ | $26.4_{19}$ | $\mathbf{37.0_{10}}$ | $\mathbf{36.6_8}$ | $\mathbf{30.6_{10}}$ | $\mathbf{30.3_8}$ | $\mathbf{28.9_{14}}$ | $\mathbf{28.5_{12}}$ | $\mathbf{30.9_5}$ | $\mathbf{30.6_4}$ |

Table 8: Absent and present keyphrase performance using Greedy Search for Transformer One2Set models in a semi-supervised setup. KPD represents KPDrop. PT represents pre-training on the synthetic data (UC). PT+KPD-R or PT+KPD-A represents pre-training on UC with KPD-R or KPD-A repsectively. FT represents fine tuning or training on the low resource labelled data (LR). PT (or PT+KPD-A or PT+KPD-R) followed by "; FT" represents that the pre-training was followed by fine-tuning of the pre-trained model on LR. We bold the overall best scores. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$).

| | Inspec | | NUS | | Krapivin | | SemEval | | KP20k | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Models** | **F@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** | **F1@M** | **F1@5** |
| **Transformer One2Set (Beam Search)** | | | | | | | | | | |
| **Absent Keyphrase Performance** | | | | | | | | | | |
| PT | $0.2_0$ | $0.7_2$ | $0.2_0$ | $0.6_3$ | $0.1_0$ | $0.2_1$ | $0.2_0$ | $0.2_2$ | $0.1_0$ | $0.3_0$ |
| PT+KPD-R | $\mathbf{0.4_0}$ | $1.4_2$ | $0.5_0$ | $1.4_2$ | $0.4_0$ | $1.2_{12}$ | $0.5_1$ | $1.2_2$ | $0.3_0$ | $1.2_1$ |
| PT+KPD-A | $0.3_0$ | $0.8_1$ | $0.3_0$ | $0.4_0$ | $0.2_0$ | $0.3_1$ | $0.2_0$ | $0.3_1$ | $0.1_0$ | $0.4_0$ |
| FT | $0.2_0$ | $0.3_2$ | $0.6_0$ | $0.8_2$ | $0.3_0$ | $1.0_2$ | $0.4_1$ | $0.8_1$ | $0.3_0$ | $0.9_0$ |
| PT; FT | $0.3_0$ | $0.8_1$ | $0.5_0$ | $1.6_3$ | $0.3_0$ | $1.4_1$ | $0.4_0$ | $0.6_1$ | $0.3_0$ | $1.1_1$ |
| PT+KPD-A; FT | $0.3_0$ | $1.0_3$ | $0.5_0$ | $1.8_4$ | $0.4_0$ | $1.7_3$ | $0.4_0$ | $1.5_2$ | $0.3_0$ | $1.7_0$ |
| PT+KPD-R; FT | $\mathbf{0.4_0}$ | $\mathbf{1.7_1}$ | $\mathbf{1.0_0}$ | $\mathbf{5.4_4}$ | $\mathbf{0.7_0}$ | $\mathbf{4.5_2}$ | $\mathbf{0.9_0}$ | $\mathbf{3.1_2}$ | $\mathbf{0.5_0}$ | $\mathbf{3.6_0}$ |
| **Present Keyphrase Performance** | | | | | | | | | | |
| PT | $19.3_8$ | $28.6_{15}$ | $16.2_3$ | $22.7_9$ | $10.3_3$ | $17.5_4$ | $18.0_9$ | $23.4_{11}$ | $10.5_4$ | $17.0_5$ |
| PT+KPD-R | $\mathbf{22.5_4}$ | $\mathbf{31.9_{15}}$ | $16.9_9$ | $23.6_7$ | $\mathbf{11.3_5}$ | $18.0_6$ | $\mathbf{19.6_{10}}$ | $22.9_{15}$ | $10.8_5$ | $17.6_7$ |
| PT+KPD-A | $20.7_3$ | $28.8_8$ | $\mathbf{17.2_4}$ | $23.5_9$ | $11.1_2$ | $17.9_{10}$ | $19.0_5$ | $22.9_{15}$ | $\mathbf{11.3_2}$ | $17.8_9$ |
| FT | $12.6_{10}$ | $11.3_5$ | $15.0_{15}$ | $23.7_9$ | $9.3_9$ | $14.2_7$ | $12.3_{10}$ | $15.3_{11}$ | $10.9_8$ | $16.8_1$ |
| PT; FT | $17.4_2$ | $24.3_8$ | $14.9_3$ | $32.1_6$ | $9.5_{12}$ | $25.0_2$ | $14.8_5$ | $27.4_{11}$ | $10.5_2$ | $25.6_3$ |
| PT+KPD-A; FT | $17.7_3$ | $25.9_2$ | $15.7_1$ | $33.2_{11}$ | $10.1_2$ | $25.8_5$ | $15.8_2$ | $26.7_{12}$ | $10.9_1$ | $26.8_2$ |
| PT+KPD-R; FT | $20.4_4$ | $\mathbf{31.9_9}$ | $15.4_2$ | $\mathbf{37.1_5}$ | $10.3_2$ | $\mathbf{29.2_7}$ | $16.6_2$ | $\mathbf{30.4_{18}}$ | $10.2_2$ | $\mathbf{29.8_1}$ |

Table 9: Absent and present keyphrase performance using Beam Search for Transformer One2Set models in a semi-supervised setup. KPD represents KPDrop. PT represents pre-training on the synthetic data (UC). PT+KPD-R or PT+KPD-A represents pre-training on UC with KPD-R or KPD-A repsectively. FT represents fine tuning or training on the low resource labelled data (LR). PT (or PT+KPD-A or PT+KPD-R) followed by "; FT" represents that the pre-training was followed by fine-tuning of the pre-trained model on LR. We bold the overall best scores. Subscripts represent standard deviation (e.g., $31.1_1$ represents $31.1 \pm 0.1$).

| Models | Inspec |  | Krapivin |  | SemEval |  | KP20k |  |
|---|---|---|---|---|---|---|---|---|
|  | **F1@M** | **F1@5C** | **F1@M** | **F1@5C** | **F1@M** | **F1@5C** | **F1@M** | **F1@5C** |
| ExHiRD Greedy | 2.2 | 1.1 | 4.3 | 2.2 | 2.5 | 1.7 | 3.2 | 1.6 |
| ExHiRD Greedy+KPD-R | **3.5** | **2.0** | **6.8** | **3.7** | **5.1** | **3.7** | **5.3** | **2.7** |
| T5 Greedy | 2.5 | 1.4 | 5.3 | 2.8 | 2.3 | 1.6 | 3.6 | 1.8 |
| T5 Greedy+KPD-R | **3.2** | **1.8** | **7.0** | **4.2** | **3.8** | **2.9** | **5.7** | **3.1** |

Table 10: Absent keyphrase performance of ExHiRD and T5. We bold the best scores per block.