# Pre-trained Language Models for Keyphrase Generation: A Thorough Empirical Study

**Anonymous ACL submission**

## Abstract

Recent years have seen unprecedented growth in natural language understanding and generation research with the help of pre-trained language models (PLMs). Autoencoding and autoregressive language model pre-training are the two dominant techniques, and recent works unify them to excel on both natural language understanding and generation tasks. In this study, we aim to fill in the vacancy of an in-depth investigation of using PLMs for keyphrase extraction and generation. We focus on keyphrase extraction as sequence labeling and keyphrase generation as sequence-to-sequence generation. Our study investigates the performance of encoder-only versus encoder-decoder PLMs, the influence of pre-training domains, and using encoders and decoders of various depths. Experiment results on benchmarks in the scientific domain and the news domain show that (1) strong and resource-efficient keyphrase generation models can be built with *in-domain encoder-only* PLMs; (2) the keyphrase extraction formulation does not help the model learn to find better present keyphrases; (3) for keyphrase generation, using a *deep encoder* and a *shallow decoder* works well. Finally, we present a strong encoder-decoder model SciBART pre-trained on a large scientific corpus and demonstrate its outstanding keyphrase generation performance and advantage over state-of-the-art PLMs.

## 1 Introduction

Keyphrases are the phrases that condense salient information of a document. Because of their high information density, keyphrases have been widely used for indexing documents, linking to relevant information, or recommending products (Wu and Bolivar, 2008; Dave and Varma, 2010). Keyphrases have also functioned as important features for text summarization (Zhang et al., 2004), information retrieval (Jones and Staveley, 1999; Song et al., 2006; Kim et al., 2013; Tang et al., 2017; Boudin et al., 2020), document clustering (Hammouda et al., 2005), and text classification (Hulth and Megyesi, 2006; Wilson et al., 2005; Berend, 2011). Recently, automatic keyphrase identification is being widely studied. Given a document, a keyphrase is a *present keyphrase* if it appears as a span in the document, or an *absent keyphrase* otherwise. The task *keyphrase extraction* requires a model to extract present keyphrases. Meng et al. (2017) further introduce *keyphrase generation* where the model is required to predict both present keyphrases and absent keyphrases.

| **Document title** |
|---|
| J.F.K. Workers Moved Drugs, Authorities Say |
| **Document body** |
| Airline employees exploited weaknesses in security procedures to help a New York drug ring smuggle heroin and cocaine through Kennedy International Airport, federal authorities charged yesterday. At least 18 people have been charged, the authorities said, including seven employees of Delta Airlines, one employee of American Airlines and two who worked at the airport. |
| **Present and Absent Keyphrases** |
| smuggling, heroin, kennedy international airport |
| drug abuse and traffic, crime and criminals, cocaine and crack cocaine |

Figure 1: An example document with present and absent keyphrases, highlighted in blue and red, respectively.

With the successful application of pre-trained language models (PLMs) on various NLP tasks (Devlin et al., 2019; Brown et al., 2020; Lewis et al., 2020; Raffel et al., 2020; Conneau et al., 2020), many latest keyphrase identification studies have based their approach on PLMs. For instance, PLMs have been used for unsupervised keyphrase extraction (Sun et al., 2020; Liang et al., 2021), keyphrase extraction via sequence labeling (Sahrawat et al., 2019; Dascalu and Trăuşan-Matu, 2021), and keyphrase generation via sequence-to-sequence (seq2seq) generation (Liu et al., 2020, 2021a; Chowdhury et al., 2022; Kulkarni et al., 2022; Wu et al., 2022; Gao et al., 2022).

| |
|---|
| **Academic domain** |
| BioBERT (Lee et al., 2019), ChemBERTa (Chithrananda et al., 2020), [Bio\|CS]_RoBERTa (Gururangan et al., 2020), SciBERT (Beltagy et al., 2019), PubMedBERT (Gu et al., 2022), MatSciBERT (Gupta et al., 2021) |
| **Social domain** |
| ClinicalBERT (Alsentzer et al., 2019), FinBERT (Liu et al., 2021b), LEGAL-BERT (Chalkidis et al., 2020), JobBERT (Zhang et al., 2022), PrivBERT (Srinath et al., 2021), Sports-BERT (Srinivasan and Mashetty) |
| **Web domain** |
| Twitter-roberta (Barbieri et al., 2020), BERTweet (Nguyen et al., 2020), [News\|Reviews]_RoBERTa (Gururangan et al., 2020), HateBERT (Caselli et al., 2021) |

Figure 2: Domain-specific encoder-only PLMs are available in a variety of domains.

Nevertheless, these studies mainly use PLMs as an effective means to achieve stronger performance, without an in-depth investigation of the pre-training domain or the model architecture. Following the seq2seq formulation in Yuan et al. (2020), previous keyphrase generation studies often use encoder-decoder PLMs such as BART (Lewis et al., 2020) or T5 (Raffel et al., 2020).[1] However, we notice that there is a diverse set of encoder-only domain-specific PLMs (Figure 2) whose domain knowledge may be leveraged to build better and more data-efficient keyphrase generation models. This leaves us with the high-level question: *are we using the right PLMs in the most effective way*?

In this empirical study, we aim at carefully searching for the best methods to use PLMs for keyphrase extraction and generation. Our study covers three important factors: (1) **encoder-only vs. encoder-decoder PLMs**, (2) the **pre-training domain**, and (3) using **a deep encoder vs. a deep decoder**. Our results suggest that domain-specific PLMs are preferred for low-resource keyphrase extraction or generation. We find that the sequence generation objective does not significantly underperform sequence labeling for predicting present keyphrases, while allowing the model to generate absent keyphrases. In addition, we explore two novel approaches to fine-tune encoder-only PLMs on keyphrase generation: (1) adding a decoder initialized from an encoder-only PLM (Rothe et al., 2020) and (2) manipulating attention masks (Dong et al., 2019). Experiments on (1) show that this approach allows an in-domain BERT (Devlin et al., 2019) to outperform a BART by a large margin in low resource scenarios. For (2), we find that a strong PLM-based encoder is crucial for keyphrase generation performance, while using a PLM decoder is not required.

Finally, to fill the vacancy of seq2seq PLMs in the scientific domain, we pre-train a BART model (which we call SciBART) using the S2ORC dataset (Lo et al., 2020) and show its effectiveness in scientific keyphrase generation. Further studies reveal that SciBART outperforms KeyBART (Kulkarni et al., 2022) which is specifically pre-trained on keyphrase generation. The major contributions of this paper are summarized as follows:

1. We show that **in-domain** PLMs are extremely sample efficient to learn keyphrase generation. We find that the in-domain **encoder-only** SciBERT outperforms BART on scientific keyphrase generation.
2. We find **a deep encoder combined with a shallow decoder** greatly outperforms the reverse configuration in terms of keyphrase generation quality and inference latency.
3. We verify that present keyphrases are not easier to extract than generate.
4. We present **SciBART**. Pre-trained on unsupervised scientific papers, it achieves better scientific keyphrase generation performance than KeyBART despite being 3x smaller.

To facilitate future research, we will make the code and pre-trained models publicly available.

## 2 Methods

### 2.1 Problem Definition

We view a keyphrase example as a triple $(\mathbf{x}, \mathbf{y_p}, \mathbf{y_a})$, corresponding to the input document $\mathbf{x} = (x_1, x_2, ..., x_d)$, the set of present keyphrases $\mathbf{y_p} = \{y_{p_1}, y_{p_2}, ..., y_{p_m}\}$, and the set of absent keyphrases $\mathbf{y_a} = \{y_{a_1}, y_{a_2}, ..., y_{a_n}\}$. For both keyphrase extraction and generation, $\mathbf{x}$ is obtained by concatenating the title, a special $[sep]$ token, and the document body. Following Meng et al. (2017), each $y_{p_i} \in \mathbf{y_p}$ is a substring of $\mathbf{x}$, and each $y_{a_i} \in \mathbf{y_a}$ does not appear in $\mathbf{x}$.

Using this formulation, the **keyphrase extraction** task requires the model to predict $\mathbf{y_p}$ in any order. We use a sequence labeling formulation for keyphrase extraction with PLMs. Concretely, for each $x_i \in \mathbf{x}$, we assign a label $c_i \in \{B_{kp}, I_{kp}, O\}$ depending on $x$ being the beginning token of a present keyphrase, the subsequent token of a present keyphrase, or otherwise. The model is required to predict the label for each token. On

[1] We use seq2seq PLMs and encoder-decoder PLMs interchangeably in this paper.

the other hand, the **keyphrase generation** task requires the prediction of $\mathbf{y_p} \cup \mathbf{y_a}$. Following Yuan et al. (2020), we use a special separator token ; to join all the keyphrases as the target sequence $\mathbf{y} = (y_{p_1}$ ; ... ; $y_{p_m}$ ; $y_{a_1}$ ; ... ; $y_{a_m})$.

### 2.2 Keyphrase Extraction

For this task, we fine-tune three encoder-only PLMs: **BERT** (Devlin et al., 2019), **SciBERT** (Beltagy et al., 2019), and **RoBERTa** (Liu et al., 2019) [2]. For each model, we add a fully connected layer to project the hidden representation of every token into 3 logits. The model is trained on the cross-entropy loss. We also experiment with using Conditional Random Field (Lafferty et al., 2001) to better model the sequence-level transitions. We use **+CRF** to refer to models with this change.

### 2.3 Keyphrase Generation

#### 2.3.1 Encoder-Decoder PLMs

Using the sequence generation formulation, we directly fine-tune **BART-base** (Lewis et al., 2020), **T5-base** (Raffel et al., 2020), and **SciBART-base** (discussed in section 2.3.3). The models are trained with cross-entropy loss for generating the target sequence of concatenated keyphrases.

#### 2.3.2 Encoder-only PLMs

**Seq2seq via BERT2BERT** We construct seq2seq models by initializing the encoder and the decoder with two encoder-only PLMs. Following Rothe et al. (2020), we add the encoder-decoder attention mechanism to the decoder. The models are then fine-tuned as seq2seq models. We use five pre-trained BERT checkpoints from Turc et al. (2019) that have hidden size 768, 12 attention heads per layer, and 2, 4, 6, 8, 10 layers, respectively. **B2B-$e$+$d$** denotes a BERT2BERT model with an $e$-layer pre-trained BERT as the encoder and a $d$-layer pre-trained BERT as the decoder. We use BERT2RND (**B2R**) to denote randomly initializing the decoder and RND2BERT (**R2B**) to denote randomly initializing the encoder.

**Seq2seq via Mask Manipulation** Dong et al. (2019) propose jointly pre-training unidirectional, bidirectional, and seq2seq language modeling by controlling mask patterns. In their seq2seq setup, the input becomes $\mathbf{x}$ $[eos]$ $\mathbf{y}$. The attention mask is designed such that tokens in $\mathbf{x}$ are only allowed to attend to other tokens within $\mathbf{x}$, and that tokens in $\mathbf{y}$ are only allowed to attend to tokens on their left. Using this formulation, we fine-tune encoder-only PLMs for seq2seq keyphrase generation. Following Dong et al. (2019), we mask and randomly replace tokens from $\mathbf{y}$ and train the model on the cross-entropy loss between its reconstruction and the original sequence. We call our models **BERT-G**, **SciBERT-G**, and **RoBERTa-G**. As these PLMs are only pre-trained on bidirectional language modeling, we further experiment on an UniLMv2 model without relative position bias (Bao et al., 2020), denoted as **UniLM**.

#### 2.3.3 In-domain Encoder-Decoder PLM

Previous works such as Beltagy et al. (2019) and Gururangan et al. (2020) have established the unique advantage of domain-specific PLMs in a wide range of tasks. To fill the vacancy of encoder-decoder PLMs in the scientific domain, we pre-train a **SciBART-base** model using the S2ORC dataset (Lo et al., 2020).

**Corpus and Data Preprocessing** The S2ORC dataset contains over 100M papers from a variety of disciplines (Figure 5). We train on all the titles and abstracts to increase the coverage of different topics. After removing non-English[3] or title-only entries, we fix wrong unicode characters, remove emails and urls, and convert the text to ASCII encoding[4]. The final dataset contains 171.7M documents, or 15.4B tokens in total. We reserve 10k documents for validation and 10k documents for testing, and use the rest as training data.

**Vocabulary** Beltagy et al. (2019) suggest that using a domain-specific vocabulary is crucial to downstream in-domain fine-tuning performance. Following their approach, we build a cased BPE vocabulary in the scientific domain using the SentencePiece[5] library on the cleaned training data. We set the vocabulary size to 30K.

**Training** For the pre-training objective, we only use text infilling as introduced in Lewis et al. (2020). We mask 30% of all tokens in each example, with the spans randomly sampled from a Poisson distribution ($\lambda = 3.5$). For 10% of the spans selected to mask, we replace them with a

[2] In this paper, we use base variants of all the encoder-only models unless otherwise specified

[3] We use guess_language for language detection.

[4] We use clean-text for data cleaning.

[5] https://github.com/google/sentencepiece

random token instead of the mask token. We set the maximum sequence length to 512. The model is pre-trained for 250k steps with batch size 2048, learning rate 3e-4, 10k warm-up steps, and polynomial learning rate decay. We use the Adam optimizer for pre-training. Using 8 Nvidia A100 GPUs, the training process took approximately 8 days.

## 3 Experimental Setup

### 3.1 Benchmarks

We test the methods in the scientific and the news domain. The statistics of all the testing datasets are provided in appendix section A.

**SciKP** Meng et al. (2017) introduce KP20k, a large keyphrase generation dataset containing 500k Computer Science papers. Following their work, we train on KP20k and evaluate on the KP20k test set and four testing datasets in the Computer Science domain: Inspec (Hulth, 2003a), Krapivin (Krapivin et al., 2009), NUS (Nguyen and Kan, 2007), and SemEval (Kim et al., 2010).

**KPTimes** Gallina et al. (2019) introduce KPTimes, a keyphrase generation dataset in the news domain containing over 250k examples. We train on the KPTimes train set and report the performance on the union of KPTimes test set and the out-of-distribution test set JPTimes.

### 3.2 Baselines

We compare PLMs with the following supervised keyphrase generation baselines:

**CatSeq** (Yuan et al., 2020) is a CopyRNN (Meng et al., 2017) trained on generating keyphrases as a sequence, separated by the separator token.

**ExHiRD-h** (Chen et al., 2021) is an improved version of CatSeq, where a hierarchical decoding framework and a hard exclusion mechanism is used to reduce duplicates.

**Transformer** and **SetTrans** are proposed by Ye et al. (2021). **Transformer** is a standard 12-layer Transformer (Vaswani et al., 2017) model with copy mechanism, while **SetTrans** performs order-agnostic keyphrase generation. Using learned control codes, **SetTrans** generates each present/absent keyphrase separately in parallel.

For keyphrase extraction, we train a randomly initialized **Transformer** as the main baseline. We also include a range of traditional keyphrase extraction methods. The full list of all baselines and their performance can be found in the appendix.

### 3.3 Evaluation

We normalize each method's predictions into a sequence of present keyphrases and a sequence of absent keyphrases. For the sequence labeling approaches, we order the phrases by the position they appear in the source document to obtain the present keyphrase predictions. Then, we apply the Porter Stemmer (Porter, 1980) to the output and target phrases and remove the duplicated phrases from the output. Following Chan et al. (2019) and Chen et al. (2020), we report the macro-averaged F1@5 and F1@M scores. For all the results except the ablation studies, we train the model with three different random seeds and report the averaged scores.

### 3.4 Implementation Details

**Keyphrase Extraction** We implemented our models with Huggingface Transformers[6] and TorchCRF[7]. The models are trained for 10 epochs with early stopping. We use a learning rate of 1e-5 with linear decay and batch size 32 for most models (see appendix for all the hyperparameters). We use AdamW with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

**Keyphrase Generation** For BART and T5, we use Huggingface Transformers and train for 15 epochs with early stopping. We use learning rate 6e-5, polynomial decay, batch size 64, and the AdamW optimizer. To fine-tune SciBART, we use the Translation task provided by fairseq[8] and train for 10 epochs. We use learning rate 3e-5, polynomial decay, and the AdamW optimizer.

For BERT-G, SciBERT-G, RoBERTa-G, and UniLM, we base on Dong et al. (2019)'s implementations [9]. For most models, we train for 20k steps with batch size 128, learning rate to 1e-4, and linear decay. We set the maximum source and target length to 464 and 48, respectively. We mask 80% of the target tokens, and randomly replace an additional 10%. We use the AdamW optimizer.

We use greedy decoding for all the models. The fine-tuning experiments are run on a local GPU server with Nvidia GTX 1080 Ti and RTX 2080 Ti GPUs. We use at most 4 GPUs and use gradient accumulation to achieve desired the batch sizes.

[6] https://github.com/huggingface/transformers

[7] https://github.com/s14t284/TorchCRF

[8] https://github.com/facebookresearch/fairseq

[9] https://github.com/microsoft/unilm

## 4 Results and Analysis

We aim to address the following questions.

1. How do PLMs compare with non-PLM approaches for rich and low resource keyphrase generation? Are encoder-decoder PLMs always better than encoder-only ones?
2. Does the keyphrase extraction formulation make present keyphrases easier to find?
3. Is a deep PLM-based encoder necessary for good keyphrase generation performance? Does the same pattern hold for the decoder?
4. How well does SciBART's performance compare with state-of-the-art PLMs?

### 4.1 In-domain PLMs are good low-resource learners for keyphrase generation

First, we compare encoder-only and encoder-decoder PLMs with established keyphrase generation methods. Table 1 presents the main results in the scientific domain. Full results are presented in the appendix. We observe that with 500k training data, directly fine-tuning base-sized PLMs on seq2seq keyphrase generation generally gives worse performance to specially designed objectives such as SetTrans, while deeper models such as T5-base can approach the state-of-the-art. However, the pattern is not consistent across domains. On KPTimes, PLMs establish an absolute advantage over CatSeq, ExHiRD, and SetTrans. We emphasize that this comparison mainly aims at faithfully examining the gap between fine-tuning PLMs and training from scratch. To build stronger models, one may consider initializing SetTrans with PLMs.

More importantly, we find the **pre-training domain** greatly influences the keyphrase generation performance of PLMs. For the scientific benchmarks, SciBERT-G and SciBART-base outperform BART-base, while on KPTimes, BART-base is stronger than T5-base despite being shallower. Moreover, **in-domain PLMs are much better low-resource learners**. From Figure 3, we clearly observe that SciBERT and SciBART require much less data to achieve as good performance as BART on KP20k, consistently outperforming the baselines using training sets of different sizes.

In summary, we highlight that (1) in rich-resource keyphrase generation, deep seq2seq PLMs can approach SOTA performance and (2) in the low-resource regime, in-domain PLMs are extremely data-efficient and greatly outperform SOTA methods trained from scratch.

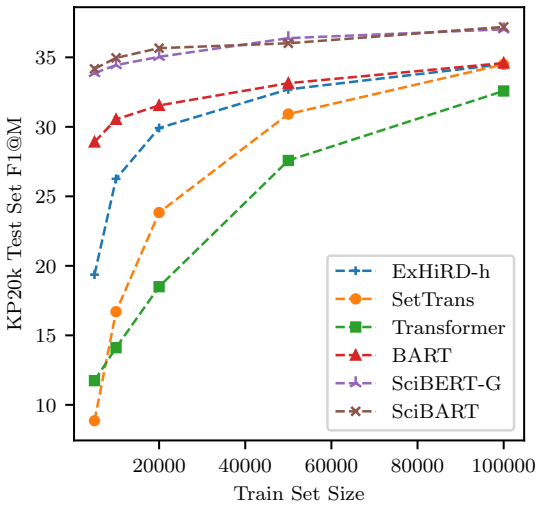


Figure 3: Different methods' present keyphrase generation performance as a function of training set size.

### 4.2 Do encoder-only models generate better keyphrases than encoder-decoder models?

In the previous section, we have established the performance of using of encoder-only and seq2seq PLMs in the classic manner, demonstrating their unique merits compared to the baselines. Next, we focus on using encoder-only PLMs for keyphrase generation. We compare the performance of (1) directly training for sequence generation by manipulating masks (BERT-G, SciBERT-G, and RoBERTa-G), (2) the best performance of assembling BERT2BERT models with the same 12-layer budget (full results are in Table 4), (3) UniLM, and (4) BART-base. Table 2 presents the keyphrase generation results on KP20k and KPTimes.

We start with the surprising result that strong keyphrase generation models can be obtained by using seq2seq attention masks to fine-tune encoder-only PLMs. On KP20k, SciBERT-G outperforms BART on all the metrics. On KPTimes, RoBERTa-G has comparable F1@5 and better F1@M for absent keyphrase generation when compared to BART. On the other hand, although specifically pre-trained on seq2seq masks, UniLM does not outperform the domain-specific SciBERT-G and RoBERTa-G. Meanwhile, out-of-domain encoder-only PLMs (i.e., BERT-G and RoBERTa-G on KP20k, or BERT-G and SciBERT on KPTimes) still cannot outperform BART. This suggests that for keyphrase generation, **an in-domain encoder-only PLM is able to outperform a domain-general encoder-decoder PLM of a similar size**.

Next, we observe that combining two smaller-

| Method | \|M\| | KP20k | | Inspec | | Krapivin | | NUS | | SemEval | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M |
| **Present keyphrase generation** | | | | | | | | | | | |
| CatSeq | 21M | 29.1 | 36.7 | 22.5 | 26.2 | 26.9 | 35.4 | 32.3 | 39.7 | 24.2 | 28.3 |
| ExHiRD-h | 22M | 31.1 | 37.4 | 25.4 | 29.1 | 28.6 | 30.8 | - | - | 30.4 | 28.2 |
| Transformer | 98M | 33.3 | 37.6 | 28.8 | 33.3 | 31.4 | 36.5 | 37.8 | 42.9 | 28.8 | 32.1 |
| SetTrans | 98M | **35.6** | 39.1 | **29.1** | 32.8 | **33.5** | **37.5** | **39.9** | **44.6** | **32.2** | **34.2** |
| SciBERT-G | 110M | 32.8 | **39.7** | 25.7 | 31.3 | 27.2 | 33.4 | 35.8 | 41.5 | 24.7 | 28.4 |
| SciBART-base | 124M | 34.1 | 39.6 | 27.5 | 32.8 | 28.2 | 32.9 | 37.3 | 42.1 | 27.0 | 30.4 |
| BART-base | 140M | 32.2 | 38.8 | 27.0 | 32.3 | 27.0 | 33.6 | 36.6 | 42.4 | 27.1 | 32.1 |
| T5-base | 223M | 33.6 | 38.8 | 28.8 | **33.9** | 30.2 | 35.0 | 38.8 | 44.0 | 29.5 | 32.6 |
| **Absent keyphrase generation** | | | | | | | | | | | |
| CatSeq | 21M | 1.5 | 3.2 | 0.4 | 0.8 | 1.8 | 3.6 | 1.6 | 2.8 | 2.0 | 2.8 |
| ExHiRD-h | 22M | 1.6 | 2.5 | 1.1 | 1.6 | 2.2 | 3.3 | - | - | 1.6 | 2.1 |
| Transformer | 98M | 2.2 | 4.6 | 1.2 | 2.3 | 3.3 | 6.3 | 2.5 | 4.4 | 1.6 | 2.2 |
| SetTrans | 98M | **3.5** | **5.8** | **1.9** | **3.0** | **4.5** | **7.2** | **3.7** | 5.5 | **2.2** | **2.9** |
| SciBERT-G | 110M | 2.4 | 4.6 | 1.4 | 2.7 | 2.4 | 4.6 | 3.4 | **5.9** | 1.3 | 1.8 |
| SciBART-base | 124M | 2.9 | 5.2 | 1.6 | 2.8 | 3.3 | 5.4 | 3.3 | 5.3 | 1.8 | 2.2 |
| BART-base | 140M | 2.2 | 4.2 | 1.0 | 1.7 | 2.8 | 4.9 | 2.6 | 4.2 | 1.6 | 2.1 |
| T5-base | 223M | 1.7 | 3.4 | 1.1 | 2.0 | 2.3 | 4.3 | 2.7 | 5.1 | 1.4 | 2.0 |

Table 1: Scientific keyphrase generation results of keyphrase generation baselines and PLM-based methods. The best results are boldfaced. In addition, the best results across the PLM-based methods are underlined.

| Method | \|M\| | KP20k | | KPTimes | |
|---|---|---|---|---|---|
| | | F1@5 | F1@M | F1@5 | F1@M |
| **Present keyphrase generation** | | | | | |
| Encoder-only PLM | | | | | |
| BERT-G | 110M | 31.3 | 37.9 | 32.3 | 47.4 |
| SciBERT-G | 110M | **32.8** | **39.7** | 33.0 | 48.4 |
| RoBERTa-G | 125M | 28.8 | 36.9 | 33.0 | 48.2 |
| UniLM | 110M | 26.7 | 34.6 | 32.4 | 47.9 |
| B2R-8+4 | 143M | 31.2 | 37.9 | 33.2 | 48.0 |
| B2B-8+4 | 143M | 32.2 | 38.0 | 33.8 | 48.6 |
| Encoder-Decoder PLM | | | | | |
| BART-base | 140M | 32.2 | 38.8 | **35.9** | **49.9** |
| **Absent keyphrase generation** | | | | | |
| Encoder-only PLM | | | | | |
| BERT-G | 110M | 1.9 | 3.7 | 16.5 | 24.6 |
| SciBERT-G | 110M | **2.4** | **4.6** | 15.7 | 24.7 |
| RoBERTa-G | 125M | 2.0 | 3.9 | **17.1** | **25.5** |
| UniLM | 110M | 1.4 | 2.8 | 15.2 | 24.1 |
| B2R-8+4 | 143M | 2.1 | 4.1 | 16.8 | 24.7 |
| B2B-8+4 | 143M | 2.2 | 4.2 | 16.8 | 24.5 |
| Encoder-Decoder PLM | | | | | |
| BART-base | 140M | 2.2 | 4.2 | **17.1** | 24.9 |

Table 2: A comparison across encoder-only and encoder-decoder PLMs for keyphrase generation. The best results are boldfaced, and the best encoder-only PLM results are underlined.

sized BERT models and training on keyphrase generation produce better results than BERT-G despite having a similar amount of parameters. On KPTimes, the B2B model with an 8-layer encoder and a 4-layer decoder (discussed in 4.4) achieves the best present keyphrase generation performance among all encoder-only PLMs. The model also has **a lower inference latency** due to its shallow decoder structure (section B).

In conclusion, we recommend that in the absence of in-domain seq2seq PLMs, **an in-domain encoder-only PLM should be preferred over the domain-general BART.** From Figure 3, this preference is even more evident in the low-resource scenarios. SciBERT only requires 5k data To achieve the same F1@M for present keyphrase of BART fine-tuned with 100k data. On the other hand, we find that BERT2BERT models are not as resource-efficient likely due to the insertion of randomly initialized parameters for cross attention.

| Method | \|M\| | KP20k | | KPTimes | |
|---|---|---|---|---|---|
| | | F1@5 | F1@M | F1@5 | F1@M |
| Transformer | 110M | 23.5 | 33.8 | 28.8 | 42.7 |
| BERT | 110M | 27.9 | 38.9 | **34.0** | 49.3 |
| RoBERTa | 125M | 27.9 | 39.4 | 33.2 | 48.9 |
| SciBERT | 110M | **28.6** | 40.5 | 31.8 | 47.7 |
| Transformer+CRF | 110M | 24.9 | 36.4 | 28.2 | 43.2 |
| BERT+CRF | 110M | 28.0 | 40.6 | 33.9 | **49.9** |
| RoBERTa+CRF | 125M | 26.7 | 39.0 | 32.4 | 48.4 |
| SciBERT+CRF | 110M | 28.4 | **42.1** | 31.8 | 48.1 |

Table 3: Present keyphrase extraction results of PLM-based sequence labeling approaches.

### 4.3 Are Present Keyphrases Easier to Extract than Generate?

Table 3 compares PLM-based sequence labeling with the Transformer baseline. It is apparent that with pre-training, encoder-only PLMs perform better than Transformer by a large margin. However, comparing SciBERT with SciBERT-G in Table 1, we find that training on seq2seq keyphrase generation greatly improves the F1@5 with minimal harm to F1@M. In other words, **the sequence**

**labeling objective does not make the finding present keyphrases easier than sequence generation**. Thus, we recommend using the latter objective as the former also suffers from predicting too few phrases (reflected by lower F1@5) and unable to predict absent keyphrases.

For BERT and SciBERT, we find that adding a CRF layer consistently improves the performance by a small margin on KP20k. Nevertheless, this observation is not true for RoBERTa, and does not apply to KPTimes. Therefore, we recommend that (1) training on keyphrase generation rather than sequence labeling and (2) prioritizing choosing a correct base PLM over tuning the CRF layer.

### 4.4 Does the depth of encoder and decoder impact keyphrase generation?

Observing that BART-base can be outperformed by the deeper T5-base, we are interested in further investigating the depth configuration of seq2seq models for keyphrase generation. Specifically, are deep encoders and deep decoders of the same importance? If one of them is less important, then under constrained parameter budgets, we can improve the performance by initializing it with a lightweight module and the other with a deeper PLM.

To answer the question, we conduct a series of ablation studies on KP20k and KPTimes. We fix a total budget of 12 layers with hidden size 768, and experiment with five different encoder-decoder combinations. Table 4 presents the results. For both datasets, we find that the performance increases sharply then plateaus as the depth of the encoder increases. With the same budget, ***a deep encoder followed by a shallow decoder*** **is strongly preferred over** ***a shallow encoder followed by a deep decoder***. We hypothesize this pattern reflects the nature of keyphrase generation: comprehending the input article is important and difficult, while generating a short string comprising several phrases based on the encoded document does not require significant knowledge from PLMs.

To verify, we further conduct the following two ablation studies. We randomly initialize either the encoder ("R2B") or the decoder ("B2R"), and train in the same manner as B2B. The results are shown in Table 4. For both datasets, we observe that **randomly initializing the encoder greatly harms the performance**, while **randomly initializing the decoder does not significantly impact the performance** (the absent keyphrase generation is even benefited in some cases). We also use keyphrase extraction to measure the learned encoder representations and the results (presented in appendix section C) align with our main findings. To summarize, we find that having a deep PLM as the encoder is important for keyphrase generation, while the initialization of the decoder is not crucial.

| $e$-$d$ | \|M\| | Arch. | KP20k | | KPTimes | |
|---|---|---|---|---|---|---|
| | | | F1@5 | F1@M | F1@5 | F1@M |
| **Present keyphrase generation** | | | | | | |
| 2-10 | 158M | B2B | 30.4 | 36.4 | 31.6 | 46.5 |
| 4-8 | 153M | B2B | 31.7 | 37.7 | 32.9 | 47.6 |
| | | R2B | 26.3 | 35.2 | 28.2 | 43.3 |
| | | B2R | 31.7 | 37.9 | 32.6 | 47.5 |
| 6-6 | 148M | B2B | 32.1 | 37.7 | **33.8** | 48.4 |
| | | R2B | 26.4 | 35.3 | 27.8 | 42.9 |
| | | B2R | 32.0 | 38.4 | 33.3 | 48.2 |
| 8-4 | 143M | B2B | **32.2** | **38.0** | **33.8** | **48.6** |
| | | R2B | 27.3 | 35.4 | 27.8 | 42.8 |
| | | B2R | 31.2 | 37.9 | 33.2 | 48.0 |
| 10-2 | 139M | B2B | 31.7 | 38.0 | 33.5 | 48.4 |
| **Absent keyphrase generation** | | | | | | |
| 2-10 | 158M | B2B | 2.1 | 3.9 | 16.2 | 23.2 |
| 4-8 | 153M | B2B | **2.2** | 4.1 | 15.9 | 23.6 |
| | | R2B | 2.5 | 4.2 | 14.7 | 24.3 |
| | | B2R | 2.2 | 4.2 | 16.5 | 24.1 |
| 6-6 | 148M | B2B | **2.2** | 4.1 | 16.4 | 24.1 |
| | | R2B | 2.6 | 4.3 | 14.5 | 20.8 |
| | | B2R | 2.3 | 4.4 | 16.2 | 23.9 |
| 8-4 | 143M | B2B | **2.2** | **4.2** | **16.8** | **24.5** |
| | | R2B | 2.4 | 4.1 | 14.9 | 21.0 |
| | | B2R | 2.1 | 4.1 | 16.8 | 24.7 |
| 10-2 | 139M | B2B | 2.1 | 4.1 | **16.8** | **24.5** |

Table 4: A comparison between different BERT2BERT architectures. In $e$-$d$, $e$ and $d$ indicates the number of encoder and decoder layers, respectively. All B2B lines are repeated with three different seeds. The best results among B2B models are boldfaced. All ablation studies (R2B and B2R) are run once with the same seed.

### 4.5 Does task-specific pre-training waive the need for in-domain pre-training?

Table 1 and Figure 3 have well demonstrated the effectiveness of SciBART on keyphrase generation. KeyBART (Kulkarni et al., 2022) is a recent effective approach by fine-tuning BART-large on keyphrase generation using the OAGKX dataset (Çano and Bojar, 2020) with a special objective that corrupts the input text by removing keyphrases. Compared to KeyBART, SciBART only performs task-agnostic in-domain pre-training. To compare and understand the effectiveness of these two training schemes, we fine-tune SciBART on keyphrase generation using OAGKX without corrupting the input text, and evaluate the resulting model's zero-shot and transfer performance on KP20k. We train

| Model | \|M\| | Present KPs | | Absent KPs | |
|---|---|---|---|---|---|
| | | F1@5 | F1@M | F1@5 | F1@M |
| KeyBART | 406M | 20.4 | 22.8 | 1.7 | 0.9 |
| KeyBART+ft | 406M | 32.5 | 39.8 | 2.6 | 4.7 |
| SciBART | 124M | 26.6 | 31.2 | 1.5 | 2.6 |
| SciBART+ft | 124M | **35.3** | **41.5** | 2.8 | **5.2** |
| SciBART$^{\dagger}$+ft | 124M | 34.1 | 39.6 | **2.9** | **5.2** |

Table 5: Comparison between SciBART and KeyBART in zero-shot and fine-tuned settings. Both KeyBART and SciBART are first trained on OAGKX to learn to generate keyphrases. "+ft" means fine-tuned on KP20k. $^{\dagger}$ indicates directly fine-tuning the pre-trained SciBART on KP20k without training on OAGKX.

SciBART with batch size 256, learning rate 3e-5, and 250k steps in total, which is approximately 2.8 epochs, comparable to Kulkarni et al. (2022) where the model is trained for 2 epochs.

The results are presented in Table 5. Despite being a 3x smaller base-sized model, utilizing its in-domain knowledge, SciBART outperforms KeyBART on the zero-shot transfer from OAGKX to KP20k. After fine-tuning on KP20k, SciBART trained on OAGK also has superior performance than KeyBART. This suggests that in-domain general pre-training followed by fine-tuning is at least as effective as the task-specific pre-training approaches used on e.g., KeyBART. In other words, we argue that for keyphrase generation, **in-domain language modeling pre-training is fundamental and can add significant value even if it precedes a large-scale task-specific pre-training**.

## 5 Related Work

**Keyphrase Extraction** Early work on keyphrase extraction mainly follow a pipelined approach. First, a range of candidates (usually noun phrases) is selected by e.g., regular expression matching. Then, various scoring methods are used to rank the candidates, and the ones with the highest scores are returned as keyphrase predictions (Hulth, 2003b; Mihalcea and Tarau, 2004; Wan and Xiao, 2008b; Bougouin et al., 2013; Sun et al., 2020; Boudin, 2018; Liang et al., 2021). Later works adopt the sequence labeling formulation, which removes the need for selecting candidates (Zhang et al., 2016; Luan et al., 2017; Sahrawat et al., 2019).

**Keyphrase Generation** Meng et al. (2017) proposes the task Deep Keyphrase Generation and a strong baseline model CopyRNN. Following works improve the architecture by adding correlation constraints (Chen et al., 2018) and linguistic constraints (Zhao and Zhang, 2019), exploiting learning signal from titles (Ye and Wang, 2018; Chen et al., 2019b), and hierarchical modeling the phrases and words (Chen et al., 2020). Yuan et al. (2020) reformulate the problem as generating a sequence of keyphrases, while Ye et al. (2021) further uses a set generation formulation to remove the influence of ordering. Other works include incorporating reinforcement learning (Chan et al., 2019; Luo et al., 2021), GANs (Swaminathan et al., 2020), and unifying keyphrase extraction with keyphrase generation (Chen et al., 2019a; Ahmad et al., 2021). Meng et al. (2021) conducts an empirical study on architecture, generalizability, phrase ordering, and decoding strategies.

More recently, Sahrawat et al. (2019), Liu et al. (2020), Liu et al. (2021a), and Dascalu and Trăuşan-Matu (2021) have considered using pre-trained BERT (Devlin et al., 2019) for keyphrase extraction and generation. In addition, Chowdhury et al. (2022), Kulkarni et al. (2022), Wu et al. (2022), and Gao et al. (2022) use seq2seq PLMs such as BART or T5 in their approach. Wu et al. (2022) show that although outperformed by task-specific models in rich resource settings, the knowledge gained in pre-training can benefit low-resource keyphrase generation. Kulkarni et al. (2022) use keyphrase generation as a pre-training task to learn rich representations.

## 6 Conclusion

In this paper, we present an empirical study of using various types of PLMs for keyphrase extraction and generation. We investigate the performance of encoder-only vs. encoder-decoder PLMs, the influence of pre-training domains, and the significance of using deeper encoder vs. decoder. We show that we can build strong and data-efficient keyphrase generation models with in-domain encoder-only PLMs. We demonstrate that the encoder has a more important role and a *deep encoder shallow decoder approach* empirically works well. Finally, we introduce a strong encoder-decoder PLM SciBART pre-trained on a large scientific corpus and show its advantage over state-of-the-art PLMs. A comparison with KeyBART suggests that task-specific pre-training does not waive the need for in-domain pre-training. Future studies may investigate PLMs with more parameters, keyphrase generation in other domains, and employing SciBART in other downstream NLP applications.

## Limitations and Ethical Statement

Due to the constraints on computational power, we do not study large language models that have more parameters than BART-large. We hope future work can continue studying the effect of scaling up model size. In addition, our study only covers two important domains for keyphrase generation. It is interesting to see whether our results can further generalize to more domains. Finally, although we have tested SciBART thoroughly on keyphrase generation, we do not study it on other NLP tasks.

S2ORC and OAGKX are released under the Creative Commons By 4.0 License. We perform text cleaning as well as email and url filtering on S2ORC to remove sensitive information, and we keep OAGKX as-is. We use the SciKP and KP-Times benchmarking datasets distributed by the original authors. No additional preprocessing is performed before fine-tuning except lower-casing and tokenization. We do not distribute any of the pre-training and benchmark datasets.

Potential risks of SciBART include accidental leakage of (1) sensitive personal information and (2) inaccurate factual information. For (1), we carefully preprocess the data in the preprocessing stage to remove personal information including emails and urls. However, we had difficulties in desensitizing names and phone numbers in the text because they have large overlap with the informative content. For (2), since SciBART is pre-trained on scientific papers, it is possible for it to generate scientific-style statements that include inaccurate information. We encourage the potential users of SciBART to not fully rely on its outputs without verifying the correctness.

Pre-training SciBART is computationally heavy and we estimate the total $CO_2$ emission to be around 150 kg using the calculation application provided by Lacoste et al. (2019). In addition, we estimate that all the fine-tuning experiments, including hyperparameter optimization, emitted around 1500 kg $CO_2$. We release the hyperparameters in the appendix section D to help the community reduce the energy spent at optimizing PLMs for keyphrase extraction and keyphrase generation.

## 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.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

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

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1162–1170, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Florian Boudin. 2016. pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 69–73, Osaka, Japan.

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, 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.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551, Nagoya, Japan. Asian Federation of Natural Language Processing.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer.

Erion Çano and Ondřej Bojar. 2020. Two huge title and keyword generation corpora of research articles. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6663–6671, Marseille, France. European Language Resources Association.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. 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.

Wang Chen, Piji Li, and Irwin King. 2021. A training-free and reference-free summarization evaluation metric via centrality-weighted relevance and self-referenced redundancy. 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 404–414, Online. Association for Computational Linguistics.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. 2020. Chemberta: Large-scale self-supervised pretraining for molecular property prediction.

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.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Cristian Dascalu and Ştefan Trăuşan-Matu. 2021. Experiments with contextualized word embeddings for keyphrase extraction. In *2021 23rd International Conference on Control Systems and Computer Science (CSCS)*, pages 447–452.

Kushal S. Dave and Vasudeva Varma. 2010. Pattern based keyword extraction for contextual advertising. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, page 1885–1888, New York, NY, USA. Association for Computing Machinery.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Samhaa R. El-Beltagy and Ahmed Rafea. 2010. KP-miner: Participation in SemEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 190–193, Uppsala, Sweden. Association for Computational Linguistics.

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.

Ygor Gallina, Florian Boudin, and Beatrice 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, Tokyo, Japan. Association for Computational Linguistics.

Yifan Gao, Qingyu Yin, Zheng Li, Rui Meng, Tong Zhao, Bing Yin, Irwin King, and Michael Lyu. 2022. Retrieval-augmented multilingual keyphrase generation with retriever-generator iterative training. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1233–1246, Seattle, United States. Association for Computational Linguistics.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2022. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23.

Tanishq Gupta, Mohd Zaki, N. M. Anoop Krishnan, and Mausam. 2021. Matscibert: A materials domain language model for text mining and information extraction.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Khaled Hammouda, Diego Matute, and Mohamed S. Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *International workshop on machine learning and data mining in pattern recognition*, pages 265–274.

Hulth and Anette. 2004. Textrank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, EMNLP '04, pages 404–411. Association for Computational Linguistics.

Anette Hulth. 2003a. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, page 216–223, USA. Association for Computational Linguistics.

Anette Hulth. 2003b. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Anette Hulth and Beáta B. Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, page 537–544, USA. Association for Computational Linguistics.

Steve Jones and Mark S. Staveley. 1999. Phrasier: A system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 160–167, New York, NY, USA. Association for Computing Machinery.

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.

Youngsam Kim, Munhyong Kim, Andrew Cattle, Julia Otmakhova, Suzi Park, and Hyopil Shin. 2013. Applying graph-based keyword extraction to document retrieval. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 864–868, Nagoya, Japan. Asian Federation of Natural Language Processing.

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

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.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 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, Peng Fu, and Weiping Wang. 2020. Reinforced keyphrase generation with bert-based sentence scorer. In *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 1–8.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. 2021b. Finbert: A pre-trained financial language representation model for financial text mining. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2641–2651, Copenhagen, Denmark. 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.

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.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.

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.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 40:211–218.

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(140):1–67.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Dhruva Sahrawat, Debanjan Mahata, Mayank Kulkarni, Haimin Zhang, Rakesh Gosangi, Amanda Stent, Agniv Sharma, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2019. Keyphrase extraction from scholarly articles as sequence labeling using contextualized embeddings.

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

Mukund Srinath, Shomir Wilson, and C Lee Giles. 2021. Privacy at scale: Introducing the PrivaSeer corpus of web privacy policies. 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 6829–6839, Online. Association for Computational Linguistics.

Prithvishankar Srinivasan and Santosh Mashetty. Sportsbert. `https://huggingface.co/microsoft/SportsBERT`.

Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906.

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.

Yixuan Tang, Weilong Huang, Qi Liu, Anthony K. H. Tung, Xiaoli Wang, Jisong Yang, and Beibei Zhang. 2017. Qalink: Enriching text documents with relevant q&a site contents. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xiaojun Wan and Jianguo Xiao. 2008a. CollabRank: Towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 969–976, Manchester, UK. Coling 2008 Organizing Committee.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, page 855–860. AAAI Press.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ian Witten, Gordon Paynter, Eibe Frank, Carl Gutwin, and Craig Nevill-Manning. 1999. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. [link].

Di Wu, Wasi Uddin Ahmad, Sunipa Dev, and Kai-Wei Chang. 2022. Representation learning for resource-constrained keyphrase generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Xiaoyuan Wu and Alvaro Bolivar. 2008. Keyword extraction for contextual advertisement. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, page 1195–1196, New York, NY, USA. Association for Computing Machinery.

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, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. 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.

Mike Zhang, Kristian Jensen, Sif Sonniks, and Barbara Plank. 2022. SkillSpan: Hard and soft skill extraction from English job postings. In *Proceedings of*

*the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4962–4984, Seattle, United States. Association for Computational Linguistics.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on Twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 836–845, Austin, Texas. Association for Computational Linguistics.

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2004. World wide web site summarization. *Web Intelli. and Agent Sys.*, 2(1):39–53.

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.

# Supplementary Material: Appendices

## A Dataset Statistics

Table 6 summarizes the statistics of all testing datasets we use. In addition, we present the topic distribution of the S2ORC dataset in Figure 5.

| Dataset | #Examples | #KP | %AKP | \|KP\| |
|---|---|---|---|---|
| KP20k | 20000 | 5.3 | 37.1 | 2.0 |
| Inspec | 500 | 9.8 | 26.4 | 2.5 |
| Krapivin | 400 | 5.9 | 44.3 | 2.2 |
| NUS | 211 | 11.7 | 45.6 | 2.2 |
| SemEval | 100 | 14.7 | 57.4 | 2.4 |
| KPTimes | 20000 | 5.0 | 37.8 | 2.0 |

Table 6: Test sets statistics. #KP, %AKP, and |KP| refer to the average number of keyphrases per document, the percentage of absent keyphrases, and the average number of words that each keyphrase contains.

## B Inference Speed

To quantify the inference speed of different BERT2BERT configurations, we measure and compare the inference throughput of B2B-2+10, B2B-4+8, B2B-6+6, B2B-8+4, B2B-10+2 on GPU and CPU. We use the best model trained on KP20k and test on the KP20k test set with batch size 1, no padding, and no speedup libraries. For GPU, we use a Nvidia GTX 1080 Ti card and test on the full KP20k test set. For CPU, we use a local server with 40 cores and test on the first 1000 examples from the KP20k test set. We report the averaged throughput (in example/s) across three runs in Figure 4. For both CPU and GPU, we observe that the throughput decreases significantly with deeper decoders. Our recommended B2B-8+4 configuration achieves better performance than 6+6 while being 37% faster on GPU and 11% faster on CPU.

## C Encoder Quality of B2B models

To further investigate the nature of the encoder representation after being trained in a BERT2BERT formulation on keyphrase generation, we separate the encoder's weights and use it as a feature extractor. Concretely, we fix the encoder weights, add a CRF layer on top of it, and train on keyphrase extraction via sequence labeling on KPTimes for 5 epochs. The results are summarized in Table 7. We find that the encoders of BERT2BERT keyphrase generation models indeed build a strong representation such that simply fine-tuning a linear classifier on the top can achieve non-trivial keyphrase extraction performance. Furthermore, the quality of encoder is positively related to the corresponding BERT2BERT performance.

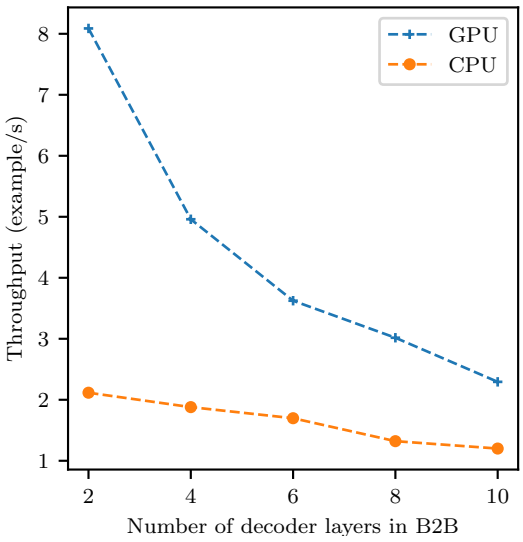


Figure 4: Inference speed of BERT2BERT models with different encoder-decoder configurations on GPU and CPU. All models have 12 layers in total. The model with x decoder layers has 12-x encoder layers.

| Model Size | \|M\| | freeze+CRF | | unfreeze | |
|---|---|---|---|---|---|
| | | F1@5 | F1@M | F1@5 | F1@M |
| 2 layers | 39M | 19.2 | 30.5 | 31.5 | 45.2 |
| 4 layers | 53M | 26.5 | 38.1 | 32.6 | 46.7 |
| 6 layers | 67M | **27.4** | **39.1** | 33.2 | 47.8 |
| 8 layers | 81M | 26.8 | 38.7 | **34.4** | **48.5** |
| 10 layers | 95M | 26.9 | 38.3 | 33.0 | 47.9 |

Table 7: Feature quality of the encoders via sequence labeling results on the KPTimes test set. The models are the encoders taken from BERT2BERT models trained on keyphrase generation and further trained on keyphrase extraction on KPTimes. "freeze" means freezing the underlying encoder model while "unfreeze" means fine-tuning the entire model. For the unfreeze version we found using CRF unnecessary.

## D Hyperparameter Optimization

For each of the PLM-based keyphrase extraction and keyphrase generation methods, we perform a careful hyperparameter search over learning rate, learning rate scheduling, batch size, and warm-up steps. The corresponding search spaces are {1e-5,

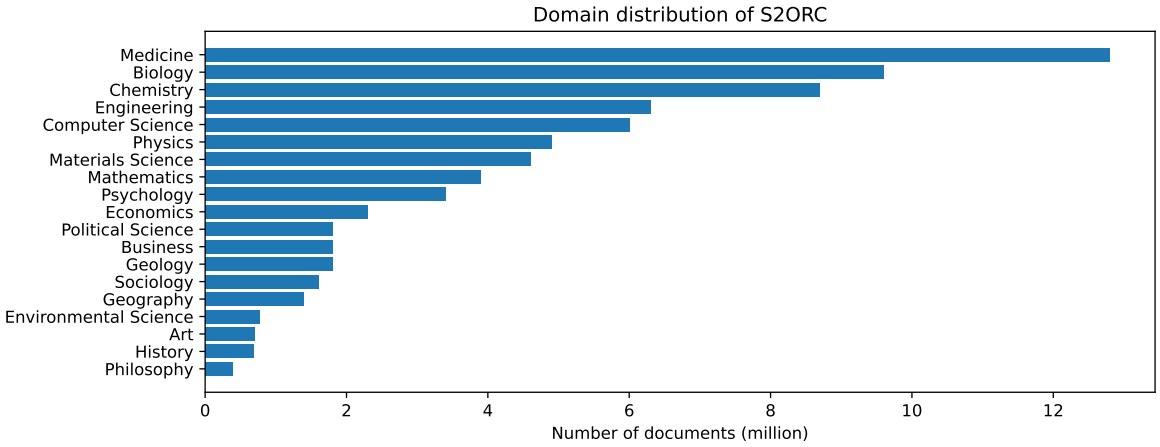


Figure 5: Domain distribution of the S2ORC dataset.

5e-4}, {linear, polynomial}, {16, 32, 64, 128}, and {500, 1000, 2000, 4000}. The best hyperparameters found are presented in Table 8.

## E Baselines and Implementation

**Keyphrase Extraction** We further compare with a range of baselines including statistical methods **YAKE** (Campos et al., 2018) and **KP-Miner** (El-Beltagy and Rafea, 2010), graph-based unsupervised methods **TextRank** (Hulth and Anette, 2004), **SingleRank** (Wan and Xiao, 2008a), **PositionRank** (Florescu and Caragea, 2017), and **MultipartiteRank** (Boudin, 2018), as well as embedding-based unsupervised methods **EmbedRank** (Bennani-Smires et al., 2018), **SIFRank+** (Sun et al., 2020), and the recent method Liang et al. (2021) which combines BERT embedding and graph structure. We also compare with a supervised feature-based model **Kea** (Witten et al., 1999). We use Boudin (2016)'s public implementations for most of these baselines. For EmbedRank and SIFRank, we use the authors' public implementations. We implement our own version of Liang et al. (2021)'s approach. We tune the hyperparameters of these method using the KP20k and KPTimes validation set.

**Keyphrase Generation** For CatSeq, we run experiments using the publicly available implementation of Chan et al. (2019).[10] For ExHiRD, Transformer, and SetTrans, we use the authors' implementations to measure the performance. We use the earliest version of KeyBART available at https://zenodo.org/record/5784384#.Y0eToNLMJcA.

## F Artifact Release

To facilitate future research, we plan to release our pre-trained SciBART checkpoint as well as the SciBART fine-tuned on OAGK. We will limit the access to the SciBART model with a non-commercial license. We will also release the raw predictions of our models to enable fair comparisons by future work.

## G All Experiment Results

We summarize all of our experiment results on SciKP and KPTimes in Table 9, Table 10, Table 11, and Table 12.

## H Qualitative Results

In Figure 6, we present a few qualitative results on KP20k from BART, T5, SciBERT, SciBART, and KeyBART.

[10] https://github.com/kenchan0226/keyphrase-generation-rl

| Model | dropout | wdecay | optimizer | bsz | #epoch | #warm-up | lr | lr schedule |
|---|---|---|---|---|---|---|---|---|
| **Keyphrase extraction** | | | | | | | | |
| Transformer | 0.1 | 0.01 | AdamW | 32 | 10 | 2000 | 3e-5 | linear |
| BERT | 0.1 | 0.01 | AdamW | 32 | 10 | 1000 | 1e-5 | linear |
| SciBERT | 0.1 | 0.01 | AdamW | 32 | 10 | 1000 | 1e-5 | linear |
| RoBERTa | 0.1 | 0.01 | AdamW | 32 | 10 | 1000 | 1e-5 | linear |
| Transformer+CRF | 0.1 | 0.01 | AdamW | 32 | 10 | 2000 | 3e-5 | linear |
| BERT+CRF | 0.1 | 0.01 | AdamW | 32 | 10 | 2000 | 1e-5 | linear |
| SciBERT+CRF | 0.1 | 0.01 | AdamW | 32 | 10 | 2000 | 1e-5 | linear |
| RoBERTa+CRF | 0.1 | 0.01 | AdamW | 32 | 10 | 2000 | 1e-5 | linear |
| **Keyphrase generation** | | | | | | | | |
| BERT-G | 0.1 | 0.01 | AdamW | 64 | 6 | 4000 | 1e-4 | linear |
| SciBERT-G | 0.1 | 0.01 | AdamW | 128 | 6 | 2000 | 1e-4 | linear |
| RoBERTa-G | 0.1 | 0.01 | AdamW | 64 | 6 | 4000 | 1e-4 | linear |
| UniLM | 0.1 | 0.01 | AdamW | 128 | 6 | 2000 | 1e-4 | linear |
| BERT2BERT | 0.0 | 0.01 | AdamW | 32 | 20 | 2000 | 5e-5 | linear |
| BART-base | 0.1 | 0.01 | AdamW | 64 | 15 | 2000 | 6e-5 | polynomial |
| SciBART-base | 0.1 | 0.01 | AdamW | 32 | 10 | 2000 | 3e-5 | polynomial |
| T5-base | 0.1 | 0.01 | AdamW | 64 | 15 | 2000 | 6e-5 | polynomial |
| KeyBART | 0.1 | 0.01 | AdamW | 64 | 15 | 2000 | 3e-5 | polynomial |

Table 8: Hyperparameters for fine-tuning PLMs for keyphrase extraction and keyphrase generation on KP20k. The hyperparameters are determined using the loss on the KP20k validation dataset. We follow a similar set of hyperparameters for KPTimes. "wdecay" = weight decay, "bsz" = batch size, "#warm-up" = number of warm-up steps, "lr" = learning rate, "lr schedule" = learning rate decay schedule. We use early stopping for all the models and use the model with lowest validation loss as the final model.

| Method | \|M\| | KP20k | | Inspec | | Krapivin | | NUS | | SemEval | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M |
| (keyphrase extraction baselines) | | | | | | | | | | | |
| KP-Miner | - | 21.9 | - | 14.4 | - | 20.6 | - | 28.3 | - | 24.4 | - |
| YAKE | - | 18.8 | - | 19.0 | - | 19.4 | - | 22.4 | - | 20.7 | - |
| TextRank | - | 16.2 | - | 22.6 | - | 13.6 | - | 20.8 | - | 18.7 | - |
| PositionRank | - | 18.9 | - | 30.4 | - | 18.9 | - | 23.0 | - | 23.8 | - |
| MultipartiteRank | - | 18.8 | - | 25.9 | - | 17.4 | - | 24.8 | - | 22.2 | - |
| EmbedRank | - | 15.5 | - | 33.6 | - | 16.9 | - | 17.3 | - | 19.2 | - |
| SIFRank+ | - | 20.0 | - | 35.1 | - | 19.6 | - | 25.5 | - | 24.8 | - |
| Liang et al. (2021) | - | 17.7 | - | 29.6 | - | 16.9 | - | 25.0 | - | 25.3 | - |
| Kea | - | 19.4 | - | 12.7 | - | 16.0 | - | 23.6 | - | 15.5 | - |
| (supervised keyphrase extraction) | | | | | | | | | | | |
| Transformer | 110M | 23.5 | 33.8 | 11.1 | 15.4 | 16.6 | 26.5 | 26.1 | 35.9 | 18.7 | 25.2 |
| Transformer+CRF | 110M | 24.9 | 36.4 | 13.3 | 18.7 | 18.9 | 29.7 | 27.8 | 37.7 | 19.8 | 27.3 |
| BERT-base | 110M | 27.9 | 38.9 | 12.8 | 17.4 | 20.7 | 30.2 | 30.9 | 41.0 | 21.8 | 28.5 |
| BERT-base+CRF | 110M | 28.0 | 40.6 | 13.7 | 18.8 | 21.0 | 32.6 | 31.3 | 41.9 | 22.3 | 29.3 |
| SciBERT | 110M | 28.6 | 40.5 | 13.1 | 17.8 | 19.9 | 30.3 | 29.7 | 39.0 | 20.0 | 26.3 |
| SciBERT+CRF | 110M | 28.4 | 42.1 | 13.9 | 19.6 | 20.6 | 32.2 | 29.9 | 40.8 | 21.3 | 28.6 |
| RoBERTa-base | 125M | 27.9 | 39.4 | 13.9 | 18.9 | 19.1 | 29.5 | 29.6 | 38.7 | 20.7 | 25.8 |
| RoBERTa-base+CRF | 125M | 26.7 | 39.0 | 12.5 | 17.5 | 18.7 | 29.3 | 28.7 | 39.5 | 20.1 | 26.8 |
| (supervised keyphrase generation) | | | | | | | | | | | |
| CatSeq | 21M | 29.1 | 36.7 | 22.5 | 26.2 | 26.9 | 35.4 | 32.3 | 39.7 | 24.2 | 28.3 |
| ExHiRD-h | 22M | 31.1 | 37.4 | 25.4 | 29.1 | 28.6 | 30.8 | - | - | 30.4 | 28.2 |
| Transformer | 98M | 33.3 | 37.6 | 28.8 | 33.3 | 31.4 | 36.5 | 37.8 | 42.9 | 28.8 | 32.1 |
| SetTrans | 98M | 35.6 | 39.1 | 29.1 | 32.8 | 33.5 | 37.5 | 39.9 | 44.6 | 32.2 | 34.2 |
| BERT-G | 110M | 31.3 | 37.9 | 25.9 | 31.3 | 26.3 | 32.2 | 35.2 | 40.9 | 26.3 | 31.0 |
| SciBERT-G | 110M | 32.8 | 39.7 | 25.7 | 31.3 | 27.2 | 33.4 | 35.8 | 41.5 | 24.7 | 28.4 |
| RoBERTa-G | 125M | 28.8 | 36.9 | 22.0 | 27.4 | 23.5 | 31.3 | 30.9 | 38.1 | 23.3 | 28.6 |
| UniLM | 110M | 26.7 | 34.6 | 18.2 | 23.6 | 23.5 | 28.5 | 28.4 | 35.3 | 21.5 | 26.8 |
| B2B-2+10 | 158M | 30.4 | 36.4 | 26.0 | 31.3 | 27.6 | 33.1 | 36.0 | 41.0 | 27.4 | 31.1 |
| B2B-4+8 | 153M | 31.7 | 37.7 | 26.5 | 31.7 | 27.1 | 32.5 | 35.6 | 40.3 | 26.0 | 30.5 |
| B2B-6+6 | 148M | 32.1 | 37.7 | 26.7 | 31.7 | 27.3 | 31.6 | 35.4 | 40.3 | 26.4 | 29.8 |
| B2B-8+4 | 143M | 32.2 | 38.0 | 26.0 | 30.9 | 27.2 | 32.1 | 36.4 | 41.8 | 28.0 | 32.8 |
| B2B-10+2 | 139M | 31.7 | 38.0 | 26.4 | 31.8 | 26.4 | 31.3 | 34.4 | 39.4 | 26.0 | 30.0 |
| SciBART-base | 124M | 34.1 | 39.6 | 27.5 | 32.8 | 28.2 | 32.9 | 37.3 | 42.1 | 27.0 | 30.4 |
| SciBART-base+OAGK | 124M | 35.3 | 41.5 | 27.1 | 33.0 | 27.7 | 33.7 | 38.2 | 42.4 | 29.2 | 32.9 |
| BART-base | 140M | 32.2 | 38.8 | 27.0 | 32.3 | 27.0 | 33.6 | 36.6 | 42.4 | 27.1 | 32.1 |
| T5-base | 223M | 33.6 | 38.8 | 28.8 | 33.9 | 30.2 | 35.0 | 38.8 | 44.0 | 29.5 | 32.6 |
| KeyBART | 406M | 32.5 | 39.8 | 26.8 | 32.5 | 28.7 | 36.5 | 37.3 | 43.0 | 26.0 | 28.9 |

Table 9: Present keyphrase evaluation results of all the methods on the SciKP benchmark. The reported results are averaged across three runs with different random seeds.

| Method | \|M\| | KP20k | | Inspec | | Krapivin | | NUS | | SemEval | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M | F1@5 | F1@M |
| CatSeq | 21M | 1.5 | 3.2 | 0.4 | 0.8 | 1.8 | 3.6 | 1.6 | 2.8 | 2.0 | 2.8 |
| ExHiRD-h | 22M | 1.6 | 2.5 | 1.1 | 1.6 | 2.2 | 3.3 | - | - | 1.6 | 2.1 |
| Transformer | 98M | 2.2 | 4.6 | 1.2 | 2.3 | 3.3 | 6.3 | 2.5 | 4.4 | 1.6 | 2.2 |
| SetTrans | 98M | 3.5 | 5.8 | 1.9 | 3.0 | 4.5 | 7.2 | 3.7 | 5.5 | 2.2 | 2.9 |
| BERT-G | 110M | 1.9 | 3.7 | 1.0 | 1.9 | 2.4 | 4.3 | 2.2 | 3.9 | 1.4 | 2.0 |
| SciBERT-G | 110M | 2.4 | 4.6 | 1.4 | 2.7 | 2.4 | 4.6 | 3.4 | 5.9 | 1.3 | 1.8 |
| RoBERTa-G | 125M | 2.0 | 3.1 | 1.0 | 2.0 | 2.7 | 4.8 | 2.5 | 4.3 | 2.1 | 2.9 |
| UniLM | 110M | 1.4 | 2.8 | 0.5 | 0.8 | 1.4 | 2.4 | 1.7 | 3.2 | 1.0 | 1.5 |
| B2B-2+10 | 158M | 2.1 | 3.9 | 1.1 | 1.9 | 2.7 | 4.7 | 2.8 | 4.7 | 1.9 | 2.6 |
| B2B-4+8 | 153M | 2.2 | 4.1 | 1.1 | 2.0 | 2.6 | 4.4 | 2.7 | 4.3 | 2.2 | 2.9 |
| B2B-6+6 | 148M | 2.2 | 4.1 | 1.0 | 1.8 | 2.7 | 4.6 | 2.8 | 4.2 | 1.7 | 2.3 |
| B2B-8+4 | 143M | 2.2 | 4.2 | 1.1 | 2.0 | 2.8 | 5.2 | 2.6 | 4.1 | 1.8 | 2.3 |
| B2B-10+2 | 139M | 2.1 | 4.1 | 1.2 | 2.3 | 2.4 | 4.4 | 2.6 | 4.6 | 1.8 | 2.5 |
| SciBART-base | 124M | 2.9 | 5.2 | 1.6 | 2.8 | 3.3 | 5.4 | 3.3 | 5.3 | 1.8 | 2.2 |
| SciBART-base+OAGK | 124M | 2.8 | 5.2 | 1.5 | 2.7 | 3.2 | 5.7 | 2.8 | 4.8 | 1.8 | 2.4 |
| BART-base | 140M | 2.2 | 4.2 | 1.0 | 1.7 | 2.8 | 4.9 | 2.6 | 4.2 | 1.6 | 2.1 |
| T5-base | 223M | 1.7 | 3.4 | 1.1 | 2.0 | 2.3 | 4.3 | 2.7 | 5.1 | 1.4 | 2.0 |
| KeyBART | 406M | 2.6 | 4.7 | 1.4 | 2.3 | 3.6 | 6.4 | 3.1 | 5.5 | 1.6 | 2.2 |

Table 10: Absent keyphrase evaluation results of all keyphrase generation methods on the SciKP benchmark.

| Method | \|M\| | F1@5 | F1@M |
|---|---|---|---|
| (keyphrase extraction baselines) | | | |
| KP-Miner | - | 18.0 | - |
| YAKE | - | 13.1 | - |
| TextRank | - | 17.4 | - |
| PositionRank | - | 11.9 | - |
| MultipartiteRank | - | 19.5 | - |
| EmbedRank | - | 10.2 | - |
| SIFRank+ | - | 15.8 | - |
| Liang et al. (2021) | - | 16.2 | - |
| Kea | - | 18.3 | - |
| (supervised keyphrase extraction) | | | |
| Transformer | 110M | 28.8 | 42.7 |
| Transformer+CRF | 110M | 28.2 | 43.2 |
| BERT-base | 110M | 34.0 | 49.3 |
| BERT-base+CRF | 110M | 33.9 | 49.9 |
| SciBERT | 110M | 31.8 | 47.7 |
| SciBERT+CRF | 110M | 31.8 | 48.1 |
| RoBERTa-base | 125M | 33.2 | 48.9 |
| RoBERTa-base+CRF | 125M | 32.4 | 48.4 |
| (supervised keyphrase generation) | | | |
| CatSeq | 21M | 29.5 | 45.3 |
| ExHiRD-h | 22M | 23.3 | 34.2 |
| Transformer | 98M | 20.2 | 34.2 |
| SetTrans | 98M | 25.9 | 37.5 |
| BERT-base-G | 110M | 32.3 | 47.4 |
| SciBERT-G | 110M | 33.0 | 48.4 |
| RoBERTa-base-G | 125M | 33.0 | 48.2 |
| UniLM | 110M | 33.2 | 48.0 |
| B2B-2+10 | 158M | 31.6 | 46.5 |
| B2B-4+8 | 153M | 32.9 | 47.6 |
| B2B-6+6 | 148M | 33.8 | 48.4 |
| B2B-8+4 | 143M | 33.8 | 48.6 |
| B2B-10+2 | 139M | 33.5 | 48.4 |
| SciBART-base | 124M | 34.8 | 48.8 |
| BART-base | 140M | 35.9 | 49.9 |
| T5-base | 223M | 34.6 | 49.2 |

Table 11: Present keyphrase evaluation results of all the methods on KPTimes. The reported results are averaged across three runs with different random seeds.

| Method | \|M\| | F1@5 | F1@M |
|---|---|---|---|
| CatSeq | 21M | 15.7 | 22.7 |
| ExHiRD-h | 22M | 7.0 | 9.1 |
| Transformer | 98M | 8.4 | 13.8 |
| SetTrans | 98M | 12.9 | 14.8 |
| BERT-G | 110M | 16.5 | 24.6 |
| SciBERT-G | 110M | 15.7 | 24.7 |
| RoBERTa-G | 125M | 17.1 | 25.5 |
| UniLM | 110M | 15.2 | 24.1 |
| B2B-2+10 | 158M | 16.2 | 23.2 |
| B2B-4+8 | 153M | 15.9 | 23.6 |
| B2B-6+6 | 148M | 16.4 | 24.1 |
| B2B-8+4 | 143M | 16.8 | 24.5 |
| B2B-10+2 | 139M | 16.8 | 24.5 |
| SciBART-base | 124M | 17.2 | 24.6 |
| BART-base | 140M | 17.1 | 24.9 |
| T5-base | 223M | 15.3 | 24.2 |

Table 12: Absent keyphrase evaluation results of all the methods on KPTimes.

**Title:** a review of design pattern mining techniques .
**Abstract:** the quality of a software system highly depends on its architectural design . high quality software systems typically apply expert design experience which has been captured as design patterns . as demonstrated solutions to recurring problems , design patterns help to reuse expert experience in software system design . they have been extensively applied in the industry . mining the instances of design patterns from the source code of software systems can assist in the understanding of the systems and the process of re engineering them . more importantly , it also helps to trace back to the original design decisions , which are typically missing in legacy systems . this paper presents a review on current techniques and tools for mining design patterns from source code or design of software systems . we classify different approaches and analyze their results in a comparative study . we also examine the disparity of the discovery results of different approaches and analyze possible reasons with some insight .
**Ground Truth:** design pattern, discovery, reverse engineering
**BART:** unrelated scheduling, mechanism design, approximation algorithms
**T5:** design patterns, software architecture, software design, software reuse
**SciBERT:** design patterns, software systems, software engineering, software mining
**SciBART:** design pattern mining, software system design, reverse engineering, software reuse
**KeyBART:** design patterns, software architecture, software reuse, software quality, software maintenance

**Title:** stabilization of second order nonholonomic systems in canonical chained form .
**Abstract:** stabilization of a class of second order nonholonomic systems in canonical chained form is investigated in this paper . first , the models of two typical second order nonholonomic systems , namely , a three link planar manipulator with the third joint unactuated , and a kinematic redundant manipulator with all joints free and driven by forces torques imposing on the end effector , are presented and converted to second order chained form by transformations of coordinate and input . a discontinuous control law is then proposed to stabilize all states of the system to the desired equilibrium point exponentially . computer simulation is given to show the effectiveness of the proposed controller .
**Ground Truth:** second order nonholonomic systems, canonical second order chained form, underactuated manipulator, discontinuous coordinate transformation, discontinuous stabilization
**BART:** stabilization, second order nonholonomic systems, canonical chained form, discontinuous control law
**T5:** stabilization, second order nonholonomic system, canonical chained form, discontinuous control
**SciBERT:** stabilization, nonholonomic system, canonical chained form, redundant manipulator
**SciBART:** stabilization, second order nonholonomic systems, canonical chained form, discontinuous control law
**KeyBART:** stabilization, second order nonholonomic systems, discontinuous control law

**Title:** characterizing output processes of e m e k [digit] queues .
**Abstract:** our goal is to study which conditions of the output process of a queue preserve the increasing failure rate ( ifr ) property in the interdeparture time . we found that the interdeparture time does not always preserve the ifr property , even if the interarrival time and service time are both erlang distributions with ifr . we give a theoretical analysis and present numerical results of e m e k [digit] queues . we show , by numerical examples , that the interdeparture time of e m e k [digit] retains the ifr property if m > k. ( c ) [digit] elsevier ltd. all rights reserved .
**Ground Truth:** ifr, erlang distribution, departure process, ph g [digit], queueing theory
**BART:** output process, increasing failure rate, interdeparture time, erlang distribution
**T5:** output process, increasing failure rate, erlang distribution
**SciBERT:** increasing failure rate, interdeparture time, erlang distribution, output process of a queue
**SciBART:** output process, increasing failure rate, interdeparture time, erlang distribution, queueing theory
**KeyBART:** output process, increasing failure rate, interdeparture time, erlang distribution, queueing theory

**Title:** optimal tool selection for 2.5 d milling , part [digit] a solid modeling approach for construction of the voronoi mountain
**Abstract:** cutter selection is a critical subtask of machining process planning . in this two part series , we develop a robust approach for the selection of an optimal set of milling cutters for a 2.5 d generalized pocket . in the first article ( part [digit] ) , we present a solid modeling approach for the construction of the voronoi mountain for the pocket geometry , which is a 3d extension of the voronoi diagram . the major contributions of this work include ( [digit] ) the development of a robust and systematic procedure for construction of the voronoi mountain for a multiply connected curvilinear polygon and ( b ) an extension of the voronoi mountain concept to handle open edges .
**Ground Truth:** 2.5 d milling, solid modelling, voronoi mountain, cutter selection, open edges
**BART:** solid modeling, voronoi mountain, cutter selection, 2.5 d generalized pocket, curve generation
**T5:** tool selection, 2.5 d milling, voronoi diagram, machining geometry
**SciBERT:** tool selection, milling, voronoi mountain, pocket geometry, cutter path planning
**SciBART:** tool selection, 2.5 d milling, voronoi mountain, cutter selection, voronoi diagram, pocket milling
**KeyBART:** tool selection, 2.5 d milling, voronoi mountain, process planning, generalized pocket, vlsi cad cam

Figure 6: Example outputs from various PLMs on the SciKP benchmarks. Correct keyphrases are colored in blue.