Keyphrase Generation: Lessons from a Reproducibility Study

Edwin Thomas[1*] and Sowmya Vajjala[2]
[1] University of Ottawa, Canada
[2]National Research Council, Ottawa, Canada
ethom123@uottawa.ca, sowmya.vajjala@nrc-cnrc.gc.ca

Abstract

Reproducibility studies are treated as means to verify the validity of a scientific method, but what else can we learn from such experiments? We addressed this question taking Keyphrase Generation (KPG) as the use case in this paper, by studying three state-of-the-art KPG models in terms of reproducibility under either the same (same data/model/code) or varied (different training data/model, but same code) conditions, and exploring different ways of comparing KPG models beyond the most commonly used evaluation measures. We drew some conclusions on the state of the art in KPG based on these experiments, and provided guidelines for researchers working on the topic about reporting experimental results in a more comprehensive manner.

Keywords: keyphrase generation, reproducibility, replicability, evaluation

1. Introduction

There has been an increasing interest in reproducibility and replicability of published results in NLP and other related areas in recent years, both in a more general, task-agnostic manner reviewing reproducibility studies across tasks, as well as in a task-specific manner, focusing on conducting relevant experiments to replicate the published results for that task (e.g., Belz et al., 2022; Arvan et al., 2022; Popović et al., 2022). The goal of such work is to understand and quantify the degree of reproducibility of a published work. The current paper takes a deeper dive into the reproducibility of approaches for one specific task - keyphrase generation, intending to understand the state of the art, as well as provide guidelines for researchers on reporting experiments more comprehensively.

Keyphrase Prediction (Extraction/Generation) is an established NLP problem, which has applications in many real-world scenarios such as information retrieval, document tagging, text summarization, and question answering, to name a few. Keyphrases are often separated into present/absent keyphrases, based on whether they directly appear in the document or not, as illustrated in Figure 1. While one strand of research on this topic focused only on present keyphrase extraction, another strand proposed methods for Keyphrase Generation (KPG), aiming to generate both present and absent keyphrases.

If each text is assigned *n* keyphrases (kps), all the KPG approaches can be grouped into three categories: One2One, where each <text, kp> pair is treated as a single training instance (e.g., Meng et al. (2017)), resulting in n instances per text; One2Seq where all n kps of a text are concatenated into a sequence with a separator between kps (e.g., Yuan et al. (2020); and One2Set, where the kps of a given text are generated as a set, instead of a concatenated sequence (e.g., Ye et al. (2021); Ray Chowdhury et al. (2022); Xie et al. (2022)). Apart from these, recent research (Martínez-Cruz et al., 2023) reported the competitive zero-shot performance of ChatGPT[1] for some KPG datasets.

Text: A *Framework to Automate the Parsing of Arabic Language Sentences This paper proposes a framework to automate the parsing (sic) of Arabic language sentences in general, although it focuses on the simple verbal sentences but it can be extended to any Arabic language sentence. The proposed system is divided into two separated phases which are lexical analysis and syntax analysis. Lexical phase analyses the words, finds its originals and roots, separates it from prefixes and suffixes, and assigns the filtered words to special tokens. Syntax analysis receives all the tokens and finds the best grammar for the given sequence of the tokens by using context free grammar. Our system assumes that the entered sentences are correct lexically and grammatically.*

Extractive: *lexical analysis, syntax analysis*

Abstractive: *Arabic language parser**

[does not appear in the text directly]*

Figure 1: An example from KP20K dataset (Meng et al., 2017) illustrating the task

Despite this reasonably large body of research on KPG and the availability of several open-access labeled datasets (mainly for English, but also for a few other languages), there is surprisingly little work on reproducibility in KPG. In comparing two approaches, there is also not much

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

[1] https://chat.openai.com/

LREC-COLING 2024, pages 9720–9731

emphasis on some of the aspects of evaluation e.g., reporting multiple evaluation measures, understanding whether two systems are more similar than they appear in terms of their predictions, performing significance testing etc. In this background, we look into reproducibility of KPG research, and study different methods of comparing two systems. Specifically, we study the following questions in this paper, considering three state-of-the-art KPG models:

1. To what extent can we reproduce existing results using the same code/model/datasets? (Section 4)
2. How does model performance vary when we use the same code/models, but train on different datasets instead? (Section 5)
3. How can we compare between two systems going beyond a single evaluation measure, and what are some good practices in reporting evaluation results? (Section 6)

We are motivated by two goals:

1. gain a better understanding of the state of the art in KPG
2. guide researchers working on the topic in terms of reporting results appropriately.

Note on the terminology: The terms reproducibility, repeatability, and replicability have been used variously and there is no consensus on their definitions. Following Belz et al. (2021)'s recommendations, we refer to all our experiments as reproduction experiments, separating them into same (where original data/model settings are retained) and varied (where training data is different) conditions instead.

The rest of this paper is organized as follows: After briefly discussing some related work (Section 2), we describe our methodology (Section 3). The next two sections focus on reproducibility under same (Section 4) and varied (Section 5) conditions, followed by a detailed analysis of approaches to compare between two systems (Section 6). Section 7 concludes the paper with a summary, followed by a discussion on the limitations and pointers to future work.

2. Related Work

Keyphrase prediction has a two-decade long history in NLP, starting from early approaches to keyword/phrase extraction (Witten et al., 1999; Mihalcea and Tarau, 2004). A lot of research focused on extractive methods, and Song et al. (2023a) provides a comprehensive overview of various approaches to present keyphrase extraction. Liu et al. (2011) first proposed to treat keyphrase prediction as machine translation, to account for absent keyphrases i.e., keyphrases that don't directly appear in the document. But Meng et al. (2017)'s application of sequence-to-sequence models for end-to-end keyphrase generation can be considered the starting point of the use of deep learning approaches that aim to generate both present and absent keyphrases together. A range of approaches including recurrent neural networks (Meng et al., 2017; Chen et al., 2018), generative adversarial networks (Swaminathan et al., 2020; Lancioni et al., 2020), reinforcement learning (Luo et al., 2021) and pre-trained language models (Wu et al., 2022a) have been explored for this task in recent years. Xie et al. (2023)'s survey presents a comprehensive survey of keyphrases generation approaches. While most of the research in this direction focuses on developing a new model for keyphrase generation, some papers address the issues around it such as dataset creation (Meng et al., 2017; Wang et al., 2019; Çano and Bojar, 2019b; Gallina et al., 2019; Yuan et al., 2020; Li et al., 2020; Piedboeuf and Langlais, 2022, etc.), and development of new evaluation measures (Chan et al., 2019; Boudin et al., 2020; Yuan et al., 2020; Luo et al., 2021; Kundu et al., 2023; Wu et al., 2023) as well.

There is some research on evaluating the state of the art and comparing different approaches to keyphrase generation in the past. Çano and Bojar (2019a)'s survey of KPG closely compared the training mechanisms of various abstractive keyphrase generation approaches proposed at that time, but does not mention any reproduction effort. Gallina et al. (2020) report on comparing multiple approaches, including reproducing two KPG models, but evaluated only the keyphrase extraction part. Meng et al. (2021) conducted a larger empirical study on how various One2One and One2Seq KPG models perform when the order of keyphrases change, comparing different decoding strategies, and reproducing some of their own previous models (Meng et al., 2017; Yuan et al., 2020). More recently, Martínez-Cruz et al. (2023) and Song et al. (2023b) compare the performance of ChatGPT's zero-shot present/absent keyphrase performance against previously reported results i.e., without reproduction. Xie et al. (2023) re-trained some of the state of the art KPG models and presented a comparison, but they also did not specifically discuss issues in reproducibilty, nor did they report results of training with multiple datasets.

In this background, we follow some previous research in terms of reproducing existing work (Gallina et al., 2020; Xie et al., 2023) and comparing decoding strategies (Meng et al., 2021) but also

extend that by training on other datasets, and, more importantly, by showing different ways to evaluate KPG and compare between two KPG approaches. In this process, we also identify some gaps in existing research and suggest guidelines for researchers to report experimental results more comprehensively.

3. Methodology

Since we aim to understand the issues related to reproducibility and evaluation, we focus on existing methods instead of proposing a new approach. We experimented with three state-of-the-art keyphrase generation approaches that shared their code. For the experiments, we employed commonly used open-access datasets reported in the research on this topic. The rest of this section describes our approach in detail.

3.1. Training Methods

We reused three publicly available implementations of recent keyphrase generation models - UniKP (Wu et al., 2021), SetTrans (Ye et al., 2021) and KPDrop (Ray Chowdhury et al., 2022), which are briefly described below:

1. UniKP jointly learns extractive and abstractive keyphrase generation using a stacked relation layer over UniLM (Dong et al., 2019), which explicitly captures the relation between extraction and abstraction[2].

2. SetTrans does not separate extraction and abstraction, and proposes to model KPG as set generation, instead of generating a sequence of keyphrases distinguished from each other by a separator.

3. KPDrop is a method to improve absent keyphrase generation in a given keyphrase generation model by randomly dropping present keyphrases and turning them into absent keyphrases during training. Of the two proposed variants in the paper (dropout and augmentation), we used augmentation method as that reported better results in the paper. We used this method on SetTrans i.e., we reproduced the One2Set+KPD-A setting from the paper.

We used the official github repositories for UniKP and KPDrop, and used KPDrop's fork of SetTrans for all our experiments[3]. By default UniKP uses beam search with a beam size of 5 for decoding whereas KPDrop and SetTrans use greedy decoding. We compare both decoding strategies later in Section 4.1.

3.2. Datasets

We used four training datasets and eight test sets (including the test splits of the four training sets), and they are all described below. For all the datasets, we followed the standard train/dev/test splits, and the recommended pre-/post-processing steps followed in the available literature on this topic.

Train The four training datasets we used in this paper were used in KPG research in the past and are described below:

1. KP20K (Meng et al., 2017) is a large dataset ($\sim$ 531k instances) of scientific paper abstracts from the computer science domain, with author assigned keyphrases, and is the most commonly used dataset to train KPG models.

2. OpenKP (Xiong et al., 2019) is a dataset consisting of real-world web documents with expert annotated keyphrases and has $\sim$148k instances. We only consider the present keyphrases part of this dataset as the number of absent keyphrases is negligible.

3. KPTimes (Gallina et al., 2019), consists of $\sim$290k news articles with expert labeled keyphrases.

4. StackEx (Yuan et al., 2020) consists of $\sim$331k questions from stackexchange.com with author assigned keyphrases.

Test Apart from the test splits of the above training datasets, we used the following commonly used four test sets.

1. Krapivin (Krapivin et al., 2009) consists of about 2000 full text computer science articles with author annotated keyphrases.

2. Inspec (Hulth, 2003) consists of 500 scientific abstracts on topics related to computer science, and is annotated by professional indexers.

3. SemEval (Kim et al., 2010) consists of 100 scientific abstracts covering computer science, material science and physics with non-author (student and expert) annotated keyphrases.

4. NUS (Nguyen and Kan, 2007) consists of 211 scientific articles with author assigned, and manually labeled keyphrases by student volunteers.

[2]The ArXiv version of the paper and ACL Anthology version have substantial differences in results. We compare with the ArXiv version as that reports results closer to the Github repo the authors shared.

[3]https://github.com/JRC1995/KPDrop/

In Section 4, we report the results for the above mentioned four test sets along with KP20k's test partition. In Section 5, we report results of training and testing with the respective partitions of the four large datasets described under Train. A summary of some basic statistics about the datasets can be found in Table 1.

Dataset	#docs	#pkp/doc	#akp/doc
Train			
KP20k	530K	2.34	2.94
KPTimes	260k	2.15	2.88
Test			
KP20k	20k	2.34	2.93
KPTimes	20k	2.72	2.31
Inspec	500	6.57	3.26
Krapivin	2304	3.73	1.6
SemEval	100	9.2	6
NUS	211	8	3.07

Table 1: Dataset Statistics (pkp and akp represent present and absent keyphrase respectively)

3.3. Evaluation Measures

Macro F1@*M*, where M is the number of predicted keyphrases (i.e., all generated keyphrases are considered to calculate F1 score) and @*k* where k=5/10/15 (where top k generated keyphrases are considered) are commonly reported for this task, by averaging the evaluation measure over all test instances, instead of micro averaging, where the averaging is done after calculating the evaluation measure per test instance. Evaluation is also reported separately for present and absent keyphrases. Among the three approaches we replicated, UniKP reports micro-F1 instead of macro-F1, though. Among other measures, F1@*O* (where O is the number of ground-truth keyphrases) is reported for both present/absent keyphrases and Recall@10/50 is reported for absent keyphrase generation. In this paper, we report macro-F1@M as the default, and discuss other evaluation measures as needed (Section 6.1). All evaluation is conducted after applying Porter stemmer (Porter, 1980) on the generated output, following the standard practice in KPG research. Detailed result tables with all evaluation measures, for all approaches/test data combinations can be found in the supplementary material[4].

3.4. Experimental setup

We conducted all our experiments on four Nvidia Tesla V100 GPUs and employed distributed training. We set most of the hyperparameters for the three systems to be the same as the respective original papers. We used a batch size of 128, distributed over four GPUs equally (per-gpu batch size of 32). While UniKP had a learning rate of 1e-05 and a warm-up proportion of 0.1, the other two approaches employed a learning rate of 1e-04 with reduction on loss plateau decay (as reported in the respective papers). To keep the training routine consistent, considering the compute resources available and the number of experiments, all the systems were trained for 5 epochs (UniKP and KPDrop were trained for 100 and 20 epochs respectively in the original papers) using the Adam Optimizer. We used the same sequence length settings as the original papers for all the approaches. For present keyphrases the maximum sequence length is set to 384 tokens for all systems. UniKP used a maximum sequence length of 40 tokens for absent keyphrases, while One2Set and KPDrop has it as 6 tokens, setting maximum number of keyphrases to 10 for both present and absent keyphrases.

4. Reproducibility: same conditions

In the first set of experiments, we focused on reproducing three KPG models using the released code and keeping the original train/test setup. Additionally, we compared the performance with greedy versus beam search decoding for these approaches.

First, we reproduced the default train/inference routine for the three approaches. Table 2 summarizes the results in terms of the difference between the reported and reproduced scores (micro-F1 for UniKP, as mentioned earlier).

Dataset	UniKP		SetTrans		KPDrop	
	Pres	Abs	Pres	Abs	Pres	Abs
KP20K	↓7.2	↓3.6	↓2.0	↓1.4	↓1.2	↓1
Krapivin	-	-	↑1	↓1.6	↓0.0	↓0.2
Inspec	↑10.2	↓0.7	↓0.2	↓0.8	↓0.5	↓1.1
SemEval	↓4.9	↓2.5	↓4	↓0.6	↓3.4	↓0.6
NUS	↓7.7	↓2.4	↓1.5	↓2.3	↓1.2	↓1.9

Table 2: Difference in performance (in percentage) while reproducing (Macro-F1@*M* for KPDrop/SetTrans; Micro-F1@*M* for UniKP)

In most cases, there was a $\pm$ 1.5% difference in the results between the reported and observed results for SetTrans and KPDrop, which can be attributed to the differences in hardware, software versions and other programming choices that are potentially unnoticed/undocumented. The drops are relatively higher for semeval's present keyphrases even for SetTrans and KPDrop (3-4%). UniKP reproduction resulted in major differences across datasets, compared to SetTrans and KPDrop. Some of it can potentially be attributed

[4] https://github.com/edwinthomas444/keyphrase-generation-reproducibility-study

to the reduced number of training epochs (5 instead of 100 used in the paper), but it is interesting to note the 10% increase in present keyphrase performance on the Inspec dataset despite that. The reduced number of training epochs can explain part of the performance drop for KPDrop and SetTrans as well (5 instead of 20 in the original papers), but clearly, the performance drops are much lesser than with UniKP. There is also a lot of variability across datasets in terms of reproducibility[5].

Since we are only reusing an existing repository and configurations, it is hard to pinpoint on specific reasons for these differences. However, these results clearly point to the difficulty of reproducibility of results even in cases where code and data processing routines are fully shared. This situation is not unique to KPG or to these repos, and is well documented in machine learning and NLP communities (Belz et al., 2021). Some recent research (Raff, 2019; Belz et al., 2022) attempts to quantify the degree of reproducibility in machine learning/NLP, which can potentially support the development of reproducible research practices that reduce the gap between the originals and reproductions in future.

4.1. Comparing decoding strategies

KPG research primarily uses two decoding strategies during inference: greedy and beam search. While UniKP and SetTrans papers report results with one default strategy, KPDrop compares beam and greedy decoding. We did not see any change in performance between greedy and beam search for UniKP (potentially because 5 epochs were not sufficient), and the results in both settings across datasets were near-identical. However, we did observe some interesting differences for SetTrans and KPDrop.

We compared two beam search settings with a low beam size (n=5) and a large beam size (n=50), following the KPDrop paper. The low beam size setting is expected to increase the overall performance by increasing the diversity of generated keyphrases, and the high beam size setting is expected to increase the recall. Table 3 and Table 4 show the performance of SetTrans and KPDrop for greedy vs beam decoding settings.

A beam size of 5 appears to be slightly better than greedy decoding in all cases for both present and absent keyphrases although they are comparable. In the larger beam size scenario, the recall does increase hugely for most cases (especially for present keyphrases), but at the cost of precision, resulting in a very large drop in F1 score in all cases (especially for absent keyphrases). Although beam search with a beam size of 5 gives the best performance across all models and datasets, it is debatable whether the slight performance improvement over greedy search is worth the increased latency during inference. Based on these results, we infer that in terms of decoding strategies, greedy and beam decoding (n=5) offer comparable performance for KPG for these models and high beam sizes are not particularly beneficial. Hence, we chose greedy search for the rest of our experiments, considering the latency time and the relatively minor performance improvement for beam search.

[5]Note: Original scores were in 30s and 40s for present keyphrases, and under 10 for absent keyphrases for all the models.

Present Keyphrases			
Dataset	Greedy	Beam=5	Beam=50
KP20K	37.16	**37.46**	13.05 (87.72)
Krapivin	36.2	**36.45**	12.88 (86.54)
Inspec	31.97	**32.91**	25.21 (82.45)
SemEval	34.31	**35.59**	21.21 (78.1)
NUS	42.2	**42.44**	19.56 (85.99)
Absent Keyphrases			
Dataset	Greedy	Beam=5	Beam=50
KP20K	4.23	**5.22**	0.37 (25.91)
Krapivin	5.16	**7.53**	0.47 (29.34)
Inspec	2.01	**2.55**	0.31 (19.07)
SemEval	2.86	**2.87**	0.53 (11.51)
NUS	4.11	**6.5**	0.57 (20.06)

Table 3: Performance of SetTrans for beam vs greedy search settings (Macro F1@M with Macro Recall@M in parantheses for Beam=50 setting)

Present Keyphrases			
Dataset	Greedy	Beam=5	Beam=50
KP20K	38.43	**38.55**	13.97 (87.66)
Krapivin	35.26	**35.77**	13.85 (86.91)
Inspec	30.06	**31.66**	27.20 (82.43)
SemEval	31.01	**31.99**	22.40 (79.82)
NUS	42.41	**43.48**	21.39 (86.31)
Absent Keyphrases			
Dataset	Greedy	Beam=5	Beam=50
KP20K	5.58	**6.27**	0.46 (28.92)
Krapivin	6.96	**7.89**	0.54 (29.04)
Inspec	2.12	**2.18**	0.37 (20.03)
SemEval	4.06	**4.73**	0.65 (13.43)
NUS	5.54	**6.53**	0.76 (24.45)

Table 4: Performance of KPDrop for beam vs greedy search settings (Macro F1@M with Macro Recall@M in parentheses for Beam=50 setting)

5. Reproducibility: other datasets

In this section, we turn to the question of how the state-of-the-art KPG approaches perform on other training datasets/domains. As described

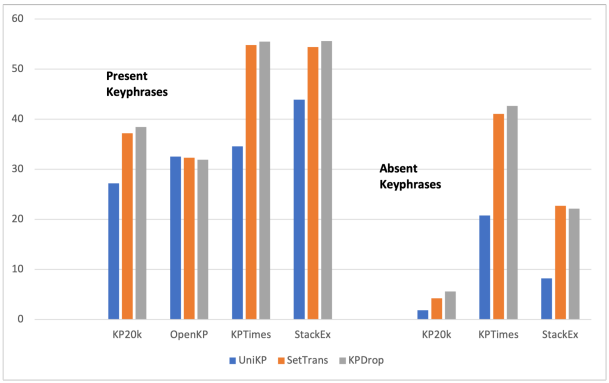


Figure 2: Performance of the three KPG models with other training datasets (Macro-F1@M)

in Section 3, we trained the three KPG methods using three other datasets (OpenKP, KPTimes, and StackEx) which come from other domains, and compared performance of present and absent keyphrase prediction (in terms of macro F1@M) in a greedy decoding setting with the models trained on KP20K in the previous section. Figure 2 shows the results.

KPDrop performs better than the other two approaches (with large improvements compared to UnikP) on 3 of the 4 the datasets for present keyphrases, with a less than 1% drop compared to UniKP on OpenKP. For absent keyphrases, KPDrop gets the best results for two of the three datasets and UniKP has the lowest scores for all the datasets. Note that the absolute performance varies widely across datasets. Studying the datasets more closely in terms of how they were created may explain the performance difference, and we leave that analysis for future work.

Cross-domain transfer: In Section 4, the four test datasets we used: Krapivin, Inspec, Semeval, NUS – all belonged to the same domain of scientific papers, with the first two test sets even sharing the sub-domain (computer science) with the training dataset KP20K. Hence, there was some transfer from the models trained on KP20K (Tables 1–3). We briefly studied the degree of cross-domain transfer with the new training datasets, taking KPDrop as the base model. There was a negligible amount of transfer for absent keyphrases and slightly better for present key phrases, but even the best result for a given train-test dataset combo hovered around 15%, which is much less than what we saw when the model was trained with KP20k.[6] This lead us to conclude that the KPG approaches we tested have some transfer to other datasets from the same domain, but there is very little transfer in out-of-domain testing scenarios. Some recent research (Meng et al., 2023) discusses approaches to improve transferability in keyphrase generation models, and it would be interesting to see if that could offer a solution for this problem in future.

[6]Detailed results are in the supplementary material.

6. Approaches to compare two KPG systems

In this section, we address the question: what are the different means to compare two KPG approaches, beyond a standard evaluation measure? We discuss the various evaluation measures used in the literature, illustrate the application of bootstrap/permutation tests for statistical significance testing for this task, and study the overlap among predictions of two systems to understand what it can tell us about KPG in general.

6.1. Evaluation Measures

Several evaluation measures are reported across papers that discuss KPG, as mentioned in Section 3. There are three issues which are important to understand a model performance appropriately and compare across models, which are discussed below:

Micro vs Macro Averaging: Calculating the averaged evaluation measure (precision/recall/f-score) over a collection of predicted keyphrases on a given test set, can be done either via macro or micro averaging. Macro averaging seems to be the most preferred approach in KPG research, although there is no clear explanation given for the choice. We assume that it is merely a convention. Some papers prefer micro-F1 instead, and some do not specify that in the paper (e.g., Wu et al. (2021, 2022b); Martínez-Cruz et al. (2023)). In this background, we explored if there are any noticeable differences between micro and macro F1 scores. Table 5 shows the average test dataset performance difference between macro and micro F1@M for KPDrop models trained on the four training sets (as mentioned before, OpenKP results are only considered for present KPs). For each model, the test sets considered for averaging include the training set's test partition, and the four other test sets (inspec, krapivin, semeval and nus).

Train	present KP	absent KP
KP20K	2.85	-0.15
KPTimes	-1.08	0.11
OpenKP	0.49	-
StackEx	-1.09	-2.21

Table 5: Difference between Macro vs Micro F1@M

The results from Table 5 indicate that macro-F1 shows larger numbers than micro-F1 for present

keyphrases for KP20K (2.85 points difference on an average) whereas micro-F1 is higher (by 2.21 points) for absent KPs for StackEx. There is no observable trend in this rise and fall across settings. Future research should consider studying this in depth, to recommend whether to report micro or macro. Until then, we would recommend researchers report macro-F1 (as that is commonly used and enables comparisons) or report both micro and macro scores, and specify what is being reported clearly.

R@10/50 for absent keyphrase prediction Some papers report R@k (where k is large e.g., 50) instead of F-scores, especially for absent keyphrase generation (Zhao and Zhang, 2019; Chen et al., 2019; Yuan et al., 2020; Meng et al., 2021). We did not find a clear motivation for this choice, although some papers speculate on potential usefulness of the measure in retrieval focused applications. However, as seen in Section 4 (Table 3 and Table 4), high recall scenarios resulted in large drops in precision. Should we prefer covering all the ground truth keyphrases (high-recall) with a lot of irrelevant keyphrases (low-precision)? Additionally, not having a common evaluation measure across present and absent keyphrases would not give us a comprehensive picture of the model performance. Hence, we would recommend reporting a common evaluation measure (an F-score), and include an additional discussion on R@k (and P@k), if needed, in main text or as supplementary material.

F@ M vs O vs K: As described earlier, KPG methods report one or two or all of the following three F1 scores: F1@M, where M is the number of generated keyphrases; F1@k, which just considers top-k generated kps and F1@O, where O is the number of ground truth keyphrases. Although F1@O is not commonly reported, it seems intuitive to expect F1@O to be a stricter measure since the generator cannot know the number of ground truth keyphrases in advance.

Table 6 shows the comparison between F1@M and F1@O using KPDrop model, trained with various datasets, and tested on the test splits of the respective datasets. The results indicate that the choice of M/O/K did not impact much when the overall performance is low (e.g., absent keyphrases in most cases, and cross-domain scenarios). While there was variation among them, it remained in the range of 0-1%. However, in scenarios where the performance was generally higher (e.g., train-test splits from the same source), there was a noticeable difference between F1@M and F1@O, but very little ($< 1\%$) difference between F1@5 and F1@M.
Clearly, these results also show that F1@O consistently assigns lower overall performance scores, which may mean that the models may be generating more key phrases than the ground truth. The usage of F1@O warrants further study and future research can perhaps focus on re-ranking the generated keyphrases to boost F1@O. Based on the results so far, it appears like F1@M and F1@5 are close to each other, and F1@O is different from both. So, we would recommend reporting F1@M and F1@O, and possibly report the rest as additional information.

Dataset	F1@M	F1@O
Present Keyphrases		
KP20k	38.43	30.85
OpenKP	31.9	27.2
StackEx	55.61	46.53
KPTimes	55.49	49.92
Absent Keyphrases		
KPTimes	42.64	37.68

Table 6: Difference between F1@M and F1@O for models with higher reported scores

6.2. Significance Testing

Conducting significance testing would help us understand whether the difference in performance between two systems is a chance variation or a statistically meaningful difference. Despite several proposed approaches in keyphrase extraction/generation literature, reporting significance is not common. Among the few who report, the usage of a paired t-test for Mean Average Precision, R@10/F@10/F@M is the most common (Boudin et al., 2020; Gallina et al., 2020). More recently, Garg et al. (2022) used a "two tailed statistical significance test" but no details are given. However, NLP literature on statistical significance has argued that t-tests are not appropriate for precision and F-score (Yeh, 2000), as normality cannot be assumed. Papagiannopoulou and Tsoumakas (2019) checks for normality to choose between t-test and wilcoxon's signed rank test on F scores for unsupervised keyphrase extraction, which seems to be a reasonable approach.

In this paper, following the recommendations of Yeh (2000) and Dror et al. (2018) on doing significance testing for NLP, we used bootstrap and permutation tests, as implemented by the latter[7]. We compared the performance of two systems that are close to each other (KPDrop and SetTrans, referenced as System A and System B respectively) on KPTimes dataset, on which both present and absent keyphrase generation had reasonable performance, in the greedy decoding setting, in

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

terms of F@5/M/O, as shown in Table 7. All the differences were statistically significant with both bootstrap and permutation tests ($p < 0.001$) except F@O-Present, where both tests indicated that the difference was not statistically significant.

	F@5	F@M	F@O
Present			
System A	55.34**	55.49**	49.92
System B	54.63	54.77	49.34
Absent			
System A	42.36**	42.64**	37.68**
System B	40.58	41.02	35.58

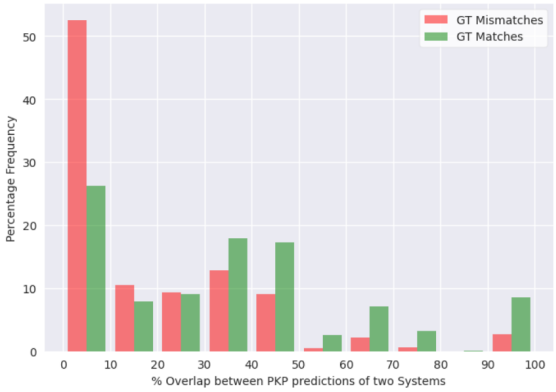
Table 7: Significance Testing across evaluation measures (** indicates that the difference between System A and B is significant ($p < 0.001$)

We did not compare UniKP with KPDrop because the performance differences were large enough to assume a significant difference (Figure 2). Clearly, the choice of evaluation measure also influences the result of the statistical significance test. Considering that the implementations of the recommended bootstrap/permutation tests are easily available, we would recommend future work on this topic to report the results of significance testing at least on results that are close to each other, and on multiple evaluation measures, if possible.

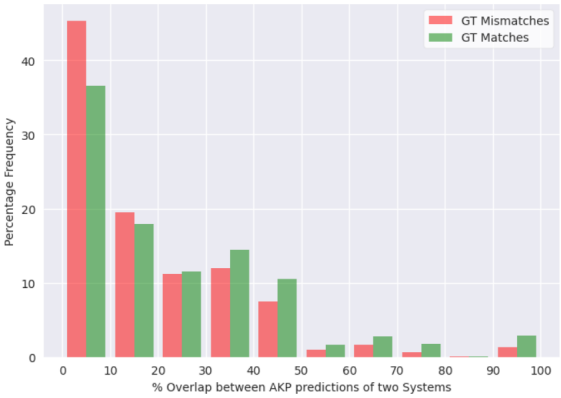
6.3. Overlap in generated output

While evaluation measures help us understand model performance, they do not give any insights into the similarities or dissimilarities between models. Observing the overlap between the generated output of two (or more) systems can be one way of addressing this. Figure 3 shows the degree of overlap between two approaches (UniKP and KPDrop) on KPTimes test set for present and absent keyphrases (PKP and AKP respectively).

From Figure 3, we observe that in the test instances where the overlap between the two systems was less (left side of the plots), there were more overlaps among the keyphrases that are not seen in the ground truth. For example, in the 0-10% overlap bracket for PKPs, over 50% of the instances that overlapped were keyphrases that did not exist in the ground truth, but were generated by both systems. This was close to 50% even in AKP. However, in the test instances where the overlap is higher (e.g., 40-50% bracket) the overlap is higher among the correctly predicted keyphrases, for both PKP and AKP. While the trend is not entirely surprising and we could expect that higher overlap would likely mean a higher overlap with ground truth too, it is interesting to notice that PKP and AKP plots are almost identical, except for the difference in the frequency of overlap.



(a)



(b)

Figure 3: Percentage Overlap between keyphrase predictions generated by UniKP and KPDrop on KPTimes test set

With regards to the high overlap among the wrong predictions, it is possible that the ground truth labels don't have a hundred percent coverage and the model generated outputs are not entirely wrong. Table 8 illustrates one example of this kind, where some extractive and abstractive keyphrases that are not seen in the ground truth are shared by two systems. The system outputs in these cases are not incorrect, but are just not covered in the ground truth. Thus, comparing overlaps across systems can potentially provide a way to expand the ground truth coverage too. Such analyses also help us understand if there are any common (or complementary) trends of success/failure across all models, and develop strategies to work around them. Future research should delve into this aspect further, devising better means of comparing the predictions of two systems, and exploring the possibility of ensemble based KPG approaches in future.

7. Conclusions

We started with the goal of understanding the state of the art in KPG research through reproducibility experiments, and looked into different ways of

Text: This paper proposes a framework to automate the parsing of Arabic language sentences in general, although it focuses on the simple verbal sentences but it can be extended to any Arabic language sentence. The proposed system is divided into two separated phases which are lexical analysis and syntax analysis. Lexical phase analyses the words, finds its originals and roots, separates it from prefixes and suffixes, and assigns the filtered words to special tokens. Syntax analysis receives all the tokens and finds the best grammar for the given sequence of the tokens by using context free grammar. Our system assumes that the entered sentences are correct lexically and grammatically.		
	Extractive	**Abstractive**
Ground Truth	lexical analysis, syntax analysis	Arabic language parser
UniKP	-	Language parsing
SetTrans	Lexical analysis; arabic language sentences; context free grammar; parsing;	arabic language parsing
KPDrop	Parsing; context free grammar ; arabic language;	arabic language parsing ; sentence retrieval ; natural language processing

Table 8: An Example showing ground truth and the predictions predictions from different models

comparing among KPG models. In terms of the research questions we started with, our learnings and some pointers to future research can be summarized as follows:

1. Reproducing results, even when code and data remain the same, is challenging. Change of hardware, training epochs etc., explain part of the difference, but not everything. More research is indeed to quantify reproducibility and delineate the factors that impact it.

2. When trained using the same code, but different training data, we noticed large differences in terms of absolute performance per dataset. This leads to a conclusion that not all keyphrase datasets are created the same, and more research is perhaps needed on the data analysis side, to understand the task better.

3. There are many evaluation measures reported in this task, and the choice of measure could affect the overall conclusions. The overlap between model predictions both when the predictions are right or wrong can give some insights into what works and doesn't work for the KPG task. So, it needs to be explored further.

Recommendations: Based on our results, we recommend the following best practices for reporting on future KPG research:

- Report the same evaluation measure for present/absent KPs, and discuss additional evaluation measures separately either in main content or supplementary material.

- Specify whether micro or macro averaging is used.

- Use the appropriate statistical significance test on close, best-performing setups.

7.1. Limitations and Outlook

We only looked at English datasets and supervised learning models in this paper, primarily to keep the number of experiments under control. We also did not repeat experiments (e.g., with different random seeds) due to computing constraints. Despite studying the overlap between systems, we did not perform any qualitative analysis yet, and designing a protocol for performing such analysis effectively for this task is an interesting direction to pursue. Finally, more experiments along the lines of Meng et al. (2023), investigating methods that achieve better transfer across domains is an important next step.

8. Ethics statement

We relied on publicly available, pre-existing code and datasets and all the code and detailed result files are provided on github[8].

Acknowledgements

We thank Rotem Dror for advice on significance testing, the three anonymous reviewers and Gabriel Bernier-Colborne, Yunli Wang and Taraka Rama for their feedback on the paper. Most importantly, we thank the authors of the papers that form the basis of this study, for sharing their code publicly, in a useable manner. This research was conducted at the National Research Council of Canada ("NRC"), thereby establishing a copyright belonging to the Crown in Right of Canada, that is, to the Government of Canada.

[8] https://github.com/edwinthomas444/keyphrase-generation-reproducibility-study/

9. References

Mohammad Arvan, Luís Pina, and Natalie Parde. 2022. Reproducibility in computational linguistics: Is source code enough? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2350–2361, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2021. A systematic review of reproducibility research in natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 381–393, Online. Association for Computational Linguistics.

Anya Belz, Maja Popovic, and Simon Mille. 2022. Quantified reproducibility assessment of NLP results. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16–28, Dublin, Ireland. 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.

Erion Çano and Ondřej Bojar. 2019a. Keyphrase generation: A multi-aspect survey. In *2019 25th Conference of Open Innovations Association (FRUCT)*, pages 85–94. IEEE.

Erion Çano and Ondřej Bojar. 2019b. Keyphrase generation: A text summarization struggle. 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 666–672, Minneapolis, Minnesota. 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, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. Title-guided encoding for keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6268–6275.

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. *Advances in neural information processing systems*, 32.

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

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

Ygor Gallina, Florian Boudin, and Béatrice Daille. 2020. Large-scale evaluation of keyphrase extraction models. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, pages 271–278.

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

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.

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.

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

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

Weidong Li, Rong Peng, Song Li, Yaqian Wang, and Zhihuan Yan. 2020. Co-occurrence graph based hierarchical neural networks for keyphrase generation. *Neurocomputing*, 415:15–26.

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

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

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

Rui Meng, Tong Wang, Xingdi Yuan, Yingbo Zhou, and Daqing He. 2023. General-to-specific transfer labeling for domain adaptable keyphrase generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1602–1618, Toronto, Canada. 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.

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

Eirini Papagiannopoulou and Grigorios Tsoumakas. 2019. Unsupervised keyphrase extraction from scientific publications. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 215–229. Springer.

Frédéric Piedboeuf and Philippe Langlais. 2022. A new dataset for multilingual keyphrase generation. *Advances in Neural Information Processing Systems*, 35:38046–38059.

Maja Popović, Sheila Castilho, Rudali Huidrom, and Anya Belz. 2022. Reproducing a manual evaluation of the simplicity of text simplification system outputs. In *Proceedings of the 15th International Conference on Natural Language Generation: Generation Challenges*, pages 80–85, Waterville, Maine, USA and virtual meeting. Association for Computational Linguistics.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Edward Raff. 2019. A step toward quantifying independently reproducible machine learning research. *Advances in Neural Information Processing Systems*, 32.

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

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

Dubrovnik, Croatia. Association for Computational Linguistics.

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

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

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.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255.

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

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

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

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

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

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

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. 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 5175–5184, Hong Kong, China. 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.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING 2000 Volume 2: The 18th International Conference on 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.

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.