An Empirical Study on Neural Keyphrase Generation

Rui Meng[♣] **Xingdi Yuan**[♠] **Tong Wang**[♠]
Sanqiang Zhao[♣] **Adam Trischler**[♠] **Daqing He**[♣]
[♣]School of Computing and Information, University of Pittsburgh
[♠]Microsoft Research, Montréal
rui.meng@pitt.edu

Abstract

Recent years have seen a flourishing of neural keyphrase generation works, including the release of several large-scale datasets and a host of new models to tackle them. Model performance on keyphrase generation tasks has increased significantly with evolving deep learning research. However, there lacks a comprehensive comparison among models, and an investigation on related factors (e.g., architectural choice, decoding strategy) that may affect a keyphrase generation system's performance. In this empirical study, we aim to fill this gap by providing extensive experimental results and analyzing the most crucial factors impacting the performance of keyphrase generation models. We hope this study can help clarify some of the uncertainties surrounding the keyphrase generation task and facilitate future research on this topic.

1 Introduction

Keyphrases are phrases that summarize and highlight important information in a piece of text. Keyphrase generation (KPG) is the task of automatically predicting such keyphrases given the source text. Typically, one source text is associated with multiple keyphrases, which may either be present in (i.e., sub-strings of) or absent from the source text.

Keyphrase generation is essentially a natural language generation (NLG) task. It has been specifically formulated as a sequence-to-sequence (Seq2Seq) problem in the existing literature (Meng et al., 2017; Chen et al., 2018; Ye and Wang, 2018; Chen et al., 2019b; Yuan et al., 2020; Chan et al., 2019; Zhao and Zhang, 2019; Chen et al., 2019a; Sun et al., 2019; Ye and Wang, 2018). Seq2Seq models are encoder-decoder neural networks, where an encoder reads through the source text to form a hidden representation, and a decoder then generates a target sequence (keyphrases) word by word conditioned on the source text representation passed by the encoder. Compared to non-neural approaches, neural models have proven to be highly effective for this task, especially in the case of predicting absent keyphrases (Meng et al., 2017).

Based on their training paradigms, most keyphrase generation models introduced in prior works fall into two categories, namely `One2One` and `One2Seq` (details to be described in Section 2). Models have achieved improved performance on texts of various types, including scientific publications (Meng et al., 2017), news articles (Gallina et al., 2019), and forum postings (Yuan et al., 2020). However, we are unaware of any existing systematic and comprehensive empirical analysis on neural keyphrase generation, particularly on examining effects of the fundamental factors shared in various model designs.

In this work, we present a comprehensive empirical study of neural keyphrase generation with extensive experiments, aiming to characterize key factors in keyphrase generation models, quantitatively analyze their impacts on model performance, and compare a wide range of baseline variants. We hope this study serves as a practical guide to help researchers on architecture, methods, and hyperparameter selection. We also hope to provide new insights to the community.

In the following sections, we first provide background regarding training paradigms, common model architectures, datasets, and evaluation metrics used in this work. We subsequently present experimental results and discussions to answer four main questions:

1. What are the pros and cons of using `One2One` vs. `One2Seq`?
2. How do different decoding strategies affect keyphrase generation?

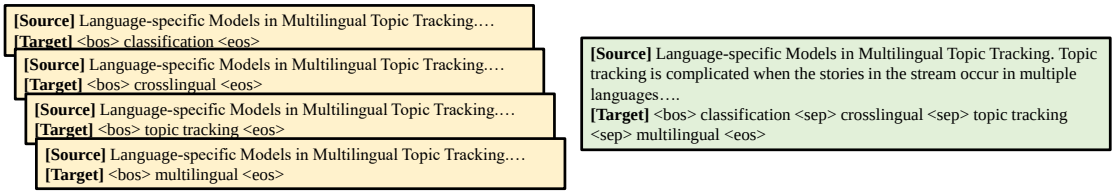

Figure 1: Comparison between One2One (left) and One2Seq (right) paradigms on the same data point.

3. Does the order of target keyphrases matter while training One2Seq?
4. Are larger models helpful? How about more training data?

2 Background

Problem Definition Formally, the task of keyphrase generation (KPG) is to generate a set of keyphrases $\{p_1, \ldots, p_n\}$ given a source text t (a sequence of words). Semantically, these phrases summarize and highlight important information contained in t, while syntactically, each keyphrase may consist of multiple words. A keyphrase is defined as *present* if it is a substring of the source text, or as *absent* otherwise.

Training Paradigms Most existing approaches for neural keyphrase generation can be categorized under one of two training paradigms: One2One (Meng et al., 2017) or One2Seq (Yuan et al., 2020), both based on the Seq2Seq (Sutskever et al., 2014) framework for NLG. Their main difference lies in how target keyphrase multiplicity is handled in constructing data points (Figure 1).

Specifically, with multiple target phrases $\{p_1, \ldots, p_n\}$, One2One takes one phrase at a time and pairs it with the source text t to form n data points $(t, p_i)_{i=1:n}$. During training, a model learns a one-to-many mapping from t to p_i's, i.e., the same source string usually has multiple corresponding target strings.

In contrast, a One2Seq system concatenates all ground-truth keyphrases p_i into a single string:

$$P = \langle \text{bos} \rangle p_1 \langle \text{sep} \rangle \cdots \langle \text{sep} \rangle p_n \langle \text{eos} \rangle,$$

(i.e., prefixed with $\langle \text{bos} \rangle$, joint with $\langle \text{sep} \rangle$, and suffixed with $\langle \text{eos} \rangle$), thus forming a single data point (t, P). A system is then trained to predict the concatenated sequence P given t.

By default in the One2Seq setting, we follow the target keyphrase ordering strategy proposed in (Yuan et al., 2020). Specifically, we sort present phrases by their first occurrences in source text, and append absent keyphrases at the end. This ordering is denoted as **PRES-ABS** in Section 5.

Architecture In this paper, we adopt the architecture used in both Meng et al. (2017) and Yuan et al. (2020), using BASERNN to denote it. BASERNN is a GRU-based Seq2Seq model (Cho et al., 2014) with a copy mechanism (Gu et al., 2016) and a coverage mechanism (See et al., 2017). Depending on different experimental settings, it is trained with either the One2One or the One2Seq paradigm. Hyperparameters are listed in Table 2.

In recent years, a host of auxiliary designs and mechanisms have been proposed and developed based on this model (see Section 7). In this study, however, we focus only on the "vanilla" version of the model and some factors we observe are important. We assume that KPG systems derived from it would be affected by these factors in similar ways.

Decoding Strategies The keyphrase generation task is distinct from other NLG tasks such as summarization and translation since it expects a set of multi-word phrases (rather than a single sequence) as model predictions. As a task that favors high recall, a common practice is to utilize beam search (Reddy et al., 1977) and take predictions from *all* beams[1], so as to proliferate the number of predicted phrases at inference time. In this work, by default we use a beam width of 200 for the One2One paradigm, and a beam width of 50 for One2Seq.

Datasets A collection of datasets in the domain of scientific publication (KP20K, INSPEC, KRAPIVIN, NUS, and SEMEVAL) and news articles (DUC) have been widely used to evaluate keyphrase generation task. Following previous work, we train models using the training set of KP20K since its size is sufficient to support the training of deep neural networks. Evaluation is performed on KP20K's test set as well as all other smaller datasets without fine-tuning on each of them. Details of the datasets are shown in Appendix A.

[1]This is in contrast to only taking the top-ranked beam as in other typical NLG tasks.

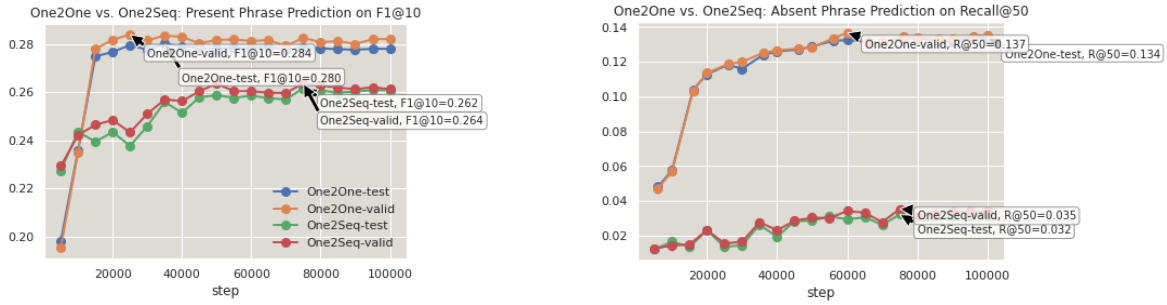

Figure 2: Models' validation and test curves on KP20K, trained with One2One and One2Seq paradigms. Left: present keyphrase prediction ($\mathbf{F}_1$**@10**); right: absent keyphrase prediction (**R@50**).

	One2One					One2Seq				
Dataset	Freq=5k	Freq=10k	Freq=20k	Freq=50k	Max Gap	Freq=5k	Freq=10k	Freq=20k	Freq=50k	Max Gap
KP20K	0.280***	0.279***	0.279***	0.278***	1.8%**	0.262	0.259	0.261	0.259	1.2%***
KRAPIVIN	0.271	0.263	0.263	0.272	3.4%**	0.271	0.265	0.272	0.265	2.6%
INSPEC	0.326***	0.324***	0.324***	0.320***	1.9%	0.384	0.382	0.387	0.382	1.3%
NUS	0.375	0.366	0.366	0.360	4.2%*	0.367	0.363	0.360	0.363	1.9%
SEMEVAL	0.344	0.352	0.352	0.344	2.3%	0.349	0.344	0.350	0.344	1.7%
DUC	0.131***	0.124***	0.124***	0.137***	10.5%**	0.157	0.159	0.159	0.159	1.3%
Out-of-distribution	0.280***	0.275***	0.275***	0.277***	1.8%*	0.302	0.299	0.303	0.299	1.2%

Table 1: Testing $\mathbf{F}_1$**@10** of One2One and One2Seq, reported according to validation scores at the frequency of {5k,10k,20k,50k} steps. Max gap is defined as $(\text{Max}(x) - \text{Min}(x))/\text{Min}(x)$ where x denotes the testing scores of four frequencies. The results of **Out-of-distribution** is acquired by concatenating the data points from five out-of-distribution testsets and compute the macro-averaged scores. */**/*** indicates the p-value of significance test (Paired Sample T-Test) is smaller than 0.05/0.01/0.001. Significance tests in "Max Gap" columns are performed between the best and worst results within four frequencies (Max(x) and Min(x)). Otherwise the significance tests are performed between corresponding results of One2One and One2Seq at the same frequency.

Evaluation Following prior studies, we evaluate present and absent keyphrases separately using $\mathbf{F}_1$**@10** and Recall@50 (**R@50**). We save model checkpoints for every 5,000 training steps and we report test performance using best checkpoint according to $\mathbf{F}_1$**@10/R@50** scores on the validation set of KP20K. Due to space limit, complete results with common metrics are included in Appendix E.

3 Generalization and Learning Dynamics

In this section, we compare the performance and learning dynamics of One2One and One2Seq. As mentioned earlier, we choose KP20K as the training data for its larger data size. In the following sections, *in-distribution* refers to the scenario where the training, validation and test sets originate from the same dataset (i.e., KP20K), whereas *out-of-distribution* refers to testing on datasets other than KP20K.

3.1 In-Distribution Performance

For each paradigm, Figure 2 shows the curves of models' performance on KP20K's validation set and test set respectively.

The first thing to notice is that One2One outperforms One2Seq by a large margin in both present and absent keyphrase prediction. We suspect that this large performance gap might be attributed to the difference on their decoding. Specifically, One2One models are capable of predicting a substantially larger number of unique phrases (more than 500, including more than 300 absent phrases). In contrast, the number of unique predictions from One2Seq models are significantly smaller (about 100, including about 50 absent phrases). Note that with the help of beam search, One2Seq models are able to generate a large number of phrases. However, in fact, only less 5% phrases are unique among all predictions. Therefore, the failure to recall enough number of unique phrases greatly restricts One2Seq models' performance on absent phrase prediction.

3.2 Out-of-Distribution Performance

We investigate One2One and One2Seq's generalizability by examining their performance and significance tests on the five transferred datasets when trained on KP20K, as shown in Table 1. Taking the

results with validation frequency of 5,000 (column Freq=5k), despite achieving significantly lower performance on KP20K, we notice that on all transferred datasets, One2Seq achieves competitive, if not better, performance compared to One2One, and this advantage is very significant on INSPEC and DUC ($p<0.001$). This suggests the interactions and dependencies among target keyphrases may facilitate generalization.

As the modern data-driven systems (e.g., deep neural networks) always eagerly require larger datasets for training, it becomes a common practice to deploy validation at a certain frequency of training steps (number of back-propagations) rather than the old-fashioned strategy of validating after every epoch. However, we observe that depending on different validation frequencies, the test performance on some keyphrase generation datasets can have non-negligible gaps. Table 1 shows that with the same model, given different validation frequency, the test performance gap can be as high as 10.5% — this is in fact larger than some of the performance gain between systems introduced in different papers. Interestingly, we find the overall gap between frequencies on One2Seq is not significant while on One2One most gaps are significant. This further reinforces our observation that One2Seq produces a better generalization performance.

This score oscillation is probably because of the small size of transferred datasets (all less than 500 data points, comparing to 20k in KP20K's test set) and raises concerns on the occasionality of the results reported on those datasets. Given the fact that researchers may not have access to the same amount of computing resources, it is less realistic to force people to use the same validation frequency. However, based on the above observations, **we suggest future work to pay attention to the validation frequency they use and report it**, thus to provide a more accurate estimation of the performance gain obtained by their proposed systems.

3.3 Convergence Speed

The curves in Figure 2 also demonstrate a large difference in convergence speed of the two paradigms. Intuitively, One2One may suffer from poor training efficiency since it splits one data example into multiple pairs, drastically increasing the number of data examples to train through the dataset. However, we observe from the curves that the One2One model converges remarkably fast with present keyphrase generation, reaching peak performance within 30k training steps before plateauing. On the other hand, without such proliferation in data examples, training One2Seq is supposedly more efficient. To our surprise, however, its performance increases steadily but slowly, and the model did not reach the optimum until after nearly 80k steps. The slower convergence speed may suggest that learning to generate multiple phrases imposes several potential challenges, including the difficulty of representing the semantics of multiple phrases, handling structural interaction among target phrases, or learning the inductive bias imposed by the concatenating order.

4 Decoding Strategies

In this section, we investigate the empirical implications of the commonly used decoding strategies — namely greedy decoding and beam search — for keyphrase generation. For beam search, we also explore to find optimal beam widths given different experiment settings.

4.1 Greedy Decoding

Since the goal of many NLG tasks is to produce one target sequence at a time, existing work often take the top-scoring beam as the output sequence. When beam width is 1, this degenerates to greedy decoding. This strategy can be (and has been) naturally applied to keyphrase generation (Yuan et al., 2020) particularly under the One2Seq paradigm. By avoiding the expansion of multiple beams, greedy decoding has obvious advantages in computational efficiency. Empirically, this decoding strategy also boasts higher precision in comparison to multi-beam decoding (to be discussed shortly). As shown in Figure 3, on the averaged test scores across all datasets, precision of greedy decoding can be up to 50–60% higher relative to the highest precision score observed in multi-beam decoding. With beam search, recall is significantly boosted, but with the sacrifice of the model's precision.

4.2 Decoding with Multiple Beams

Keyphrase generation is evaluated to match multiple keyphrases in the ground-truth. As a result, a common strategy is to take multiple beams during decoding in order to obtain more phrases. This choice is at times not only practical but in fact necessary: under the One2One paradigm, for ex-

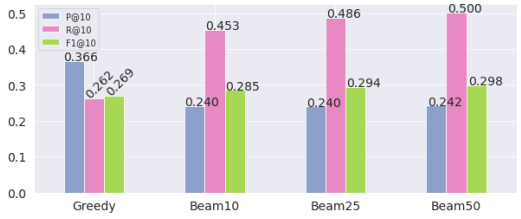

Figure 3: **P@10**, **R@10**, and **F_1@10** obtained by a `One2Seq` model using various decoding strategies. Scores are averaged across 6 datasets.

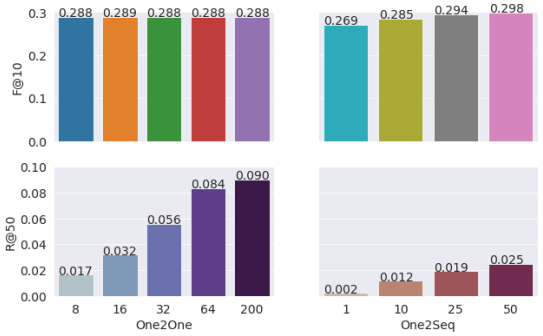

Figure 4: Models' testing performance with different beam widths (averaged across 6 datasets). Upper: present; lower: absent. Left: `One2One`; right: `One2Seq`.

ample, it is crucial to have multiple beams in order to generate multiple keyphrases for a given input.

Generally speaking, keyphrase generation is a task that strongly favors higher recall. It is thus not totally unexpected that the high precision scores of greedy decoding are often undermined by notable disadvantages in recall, which in turn leads to losing by large margins in F-scores when compared to results of beam search (with multiple beams): empirically, beam search can sometimes achieve a relative gain of more than 10% in present phrase generation (Figure 3), and a much larger performance boost in absent phrase generation (Figure 4), over greedy decoding.

We are also interested in seeing if there exists an optimal beam width. In Figure 4, we show models' testing performance when various beam widths are used. In present keyphrase generation task with `One2One` (upper left), beam width of 8 already provides an optimal score, larger beam widths (even 200) do not show any further advantage. Replacing the training paradigm with `One2Seq` (upper right), we observe a positive correlation between beam width and testing score — larger beam widths lead to marginally better testing scores. However, the improvement is not significant.

On absent keyphrase generation task (lower), both `One2One` and `One2Seq` paradigms seem to benefit from larger beam widths. Testing score shows strong positive correlation with beam width. We observe that this trend is consistent across most datasets, we provide detailed results corresponding to Figure 4 in Appendix B.

Overall, a larger beam width provides better scores in most settings, but the performance gain diminishes quickly towards very large beam width. In addition, it is worth noting that larger beam width also comes with more intense computing demands, for both space and time. As an example, in Figure 4 (left), we observe that with the `One2One` training paradigm, a beam width of 200 does not show a significant advantage over 64, however, in terms of computation, beam width of 200 takes more than 3 times of the resources compared to 64. There clearly exists a trade-off between beam width and computational efficiency (e.g., carbon footprint (Strubell et al., 2019)). We thus hope our results can serve as a reference for researchers, to help them choose beam width more wisely depending on specific tasks.

5 Does Order Matter in `One2Seq`?

In `One2One` paradigm, each data example is split to multiple equally weighted data pairs, thus it generates phrases without any prior on the order. In contrast, `One2Seq` training has the unique capability of generating a varying number of keyphrases in a single sequence. This inductive bias enables a model to learn dependencies among keyphrases, and also to implicitly estimate the number of target phrases conditioned on the source text. However, the `One2Seq` approach introduces a new complication. During training, the Seq2Seq decoder takes the concatenation of multiple target keyphrases as target. As pointed out by Vinyals et al. (2016), order matters in sequence modeling tasks; yet the ordering among the target keyphrases was not fully investigated and its effect to the models' performance remains unclear. Several studies have noted this problem (Ye and Wang, 2018; Yuan et al., 2020) without further exploration.

5.1 Ordering Definition

To explore along this direction, we first define six ordering strategies for concatenating target phrases.

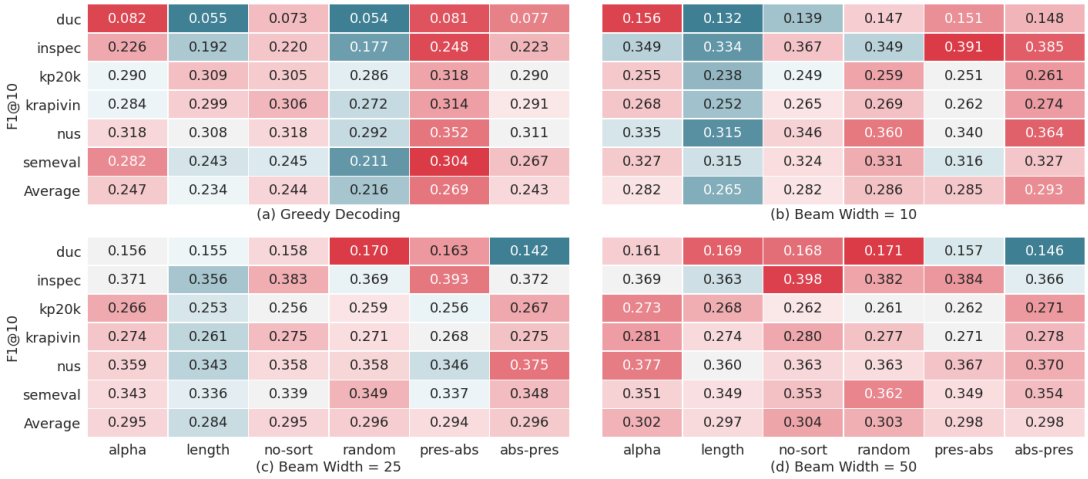

Figure 5: Present keyphrase generation testing scores. Coloring (from blue to red) represents the relative performance, from low to high, normalized per row.

RANDOM: Randomly shuffle the target phrases before every epoch. Because of the *set generation* nature of the keyphrase generation, we expect randomly shuffled target sequences help to learn an order-invariant decoder.

NO-SORT: Keep phrases in their original order in the data (e.g., provided by the authors of source texts). This was used by Ye and Wang (2018).

LENGTH: Sort phrases by their lengths (in ascending order).

ALPHA: Sort phrases by alphabetical order.

PRES-ABS: Sort present phrases by their first occurrences in source text. Absent phrases are shuffled and appended to the end of the present phrase sequence. This was used by (Yuan et al., 2020).

ABS-PRES: Similar to **PRES-ABS**, but prepending absent phrases to the beginning of present phrases.

5.2 Order Matters

Greedy Decoding In Figure 5(a), we show a model's $\mathbf{F}_1$**@10** on present keyphrase generation task, equipped with greedy decoding and each of the ordering strategies. In this setting, the model simply chooses the token with the highest probability at every step, and terminates either upon generating the end-token <EOS> or reaching the maximum target length limit (up to 40 words). This means the model predicts phrases solely relying on its innate distribution learned from the training data, and thus this performance could somewhat reflect to which degree the model fits the training distribution and understands the task.

Through this set of experiments, we first observe that each model demonstrates consistent performance across all six test datasets, indicating that ordering strategies play critical roles in training One2Seq models.

RANDOM consistently yields lower $\mathbf{F}_1$**@10** than other ordering strategies on all datasets. This suggests that a consistent order of the keyphrases is beneficial when greedy decoding is applied. Meanwhile, **PRES-ABS** outperforms other ordering strategies by significant margins. A possible explanation is that with this order (of occurrences in the source text), the current target phrase is always to the right of the previous one, which can serve as an effective prior for the attention mechanism throughout the One2Seq decoding process.

Beam Search Next, we show results obtained from the same set of models equipped with beam search in Figure 5(b/c/d). We can clearly observe the overall $\mathbf{F}_1$**@10** scores have positive correlation with the beam width (greedy decoding is a special case where beam width equals to 1). Consistent with our observations in Section 4.2, when very large beam width is used, the gain provided by beam search tends to diminish. However, compared to the greedy decoding case, the pattern among different ordering strategies appears to be less clear, with the average scores distributed more evenly across different settings.

We suspect that the uniformity among different ordering strategies with beam search may be due to the limitation of the evaluation metric $\mathbf{F}_1$**@10**. As a standard metric widely used in the existing literature, $\mathbf{F}_1$**@10** truncates a model's predictions to 10 top-ranked keyphrases. By investigation, we

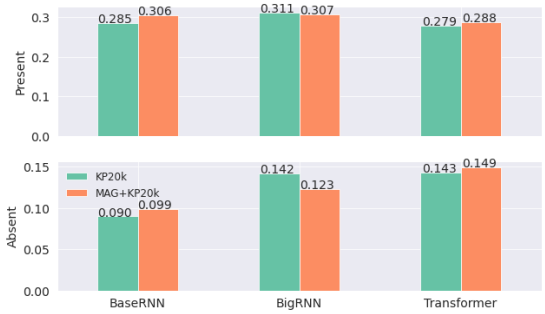

Figure 6: Model variants' averaged testing scores. Trained with One2One paradigm and augmented data.

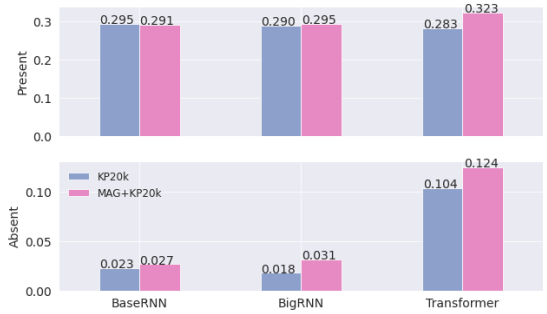

Figure 7: Model variants' averaged testing scores. Trained with One2Seq paradigm and augmented data.

find that the number of predictions by different ordering strategies varies greatly: **PRES-ABS** can generally predict more phrases than other strategies, which explains its performance advantage in greedy decoding. But as the beam width increases, all models can predict more than 10 phrases. In this case, the $\mathbf{F}_1$**@10** is contributed more by the rankings of the predictions rather than its amount. Thus the performance gap among ordering strategies is gradually narrowed in beam search (the difference between **PRES-ABS** and **LENGTH** is 0.2/0.1/0.01 when beam width is 10/25/50). The advantage of **PRES-ABS** is also clearly shown in absent keyphrase prediction (see Appendix C).

It is also worth mentioning, we observe the **ALPHA** ordering surprisingly yields satisfactory scores and training stability. We manually check the output sequences in test set, we notice that the model is actually able to retain alphabetical order among the predicted keyphrases, hinting that a Seq2Seq model might be capable of learning simple morphological dependencies even without access to any character-level representations.

6 Big Model, Big Data?

In this section, we further explore the effects of keyphrase generation performance from the perspective of model complexity and amount of training data. In addition to the aforementioned RNN-based Seq2Seq model (referred as BASERNN), we introduce two variants with much more parameters:

- BIGRNN as in (Ye and Wang, 2018), it shares the same architecture with BASERNN, except a larger embedding size and hidden size are used;
- TRANSFORMER as used in (Gehrmann et al., 2018), it is a 6-layer transformer with 8 heads, equipped with a copy attention mechanism.

We list the information of three model variants in Table 2. To offset the risk of over-fitting introduced by the additional model parameters, we adopt the MAGKP dataset (Sinha et al., 2015) to augment the training (KP20K). MAGKP is a keyphrase dataset constructed on the basis of Microsoft Academic Graph. The original dataset (v1) contains more than 166 million academic papers across different domains. We apply a filtering process by only keeping data points under the computer science topic with at least one keyphrase, resulting in about 2.7 million data points. Note that this dataset remains noisy despite the stringent filtering criteria, both because it is crawled from the web and the fact that some keywords are retrieved automatically.

Due to space limitation, scores reported in this section are the average across the 6 test dataset; more detailed results can be found in Appendix D.

First, we show the test performance of the three model variants, trained with One2One paradigm (green bars in Figure 6). On the present keyphrase generation task (upper), BIGRNN yields much better performance. TRANSFORMER produces similar test scores as BASERNN despite having 7 times more parameters.

With more training data from MAGKP (orange bars), both BASERNN and TRANSFORMER show some marginal improvement in performance, while the performance for BIGRNN actually decreases. This is observed for both present and absent keyphrase generation. Consistent with our observations in the present setting, data augmentation slightly boosts BASERNN and TRANSFORMER's performance, and has negative effect on BIGRNN.

Model	Embedding Size	Hidden Size	#Parameters
BASERNN	100	150	13M
BIGRNN	128	512	37M
TRANSFORMER	512	512	95M

Table 2: The information of three model variants.

Next, we show the same models' performance when trained with One2Seq paradigm in Figure 7. On the present keyphrase generation task (upper), model performances are quite similar for all three model variants when trained on KP20K only. With more training data, the two RNN models do not show obvious performance gain, while TRANSFORMER has a significant improvement. In absent keyphrase generation task (lower), TRANSFORMER is a clear winner, which outperforms the RNN-based models by 500%. All three models have performance boost from more training data, in which, TRANSFORMER benefits the most.

To summarize, with the same number of parameters, RNN models trained in One2One style in general yield better performance: this is particularly true for absent keyphrase generation. Meanwhile, a larger RNN-based model helps on One2One paradigm, but it does not improve performance with One2Seq, even when trained on more training data. TRANSFORMER with One2Seq, however, is able to better leverage larger datasets and produce competitive or even better test performance compared to its One2One counterparts.

It is also worth mentioning that across all experiment settings, the benefit of having larger training dataset has more significant effect on the 5 transfer learning datasets (i.e., KRAPIVIN, NUS, SEMEVAL, INSPEC, DUC), whereas on KP20K, the testing results are almost always worse than the case where no data augmentation is applied.

7 Related Work

7.1 Traditional Keyphrase Extraction

Keyphrase extraction has been studied extensively for decades. A common approach is to formulate it as a two-step process. Specifically, a system first selects a set of candidate phrases from the text. The selecting process is usually relying on heuristics and pre-defined features such as part-of-speech (POS) tags (Witten et al., 1999; Liu et al., 2011; Wang et al., 2016; Yang et al., 2017). Subsequently, a ranker is used to select the top ranked candidates following various criteria. A wide variety of methods are used as the ranker, such as bagged decision trees (Medelyan et al., 2009; Lopez and Romary, 2010), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM) (Lopez and Romary, 2010) and PageRank (Mihalcea and Tarau, 2004; Le et al., 2016; Wan and Xiao, 2008). Compared to the newly developed data driven approaches with deep neural networks, the above approaches suffer from poor performance and the need of dataset-specific heuristic design.

7.2 Neural Keyphrase Extraction

On neural keyphrase extraction task, Zhang et al. (2016); Luan et al. (2017); Gollapalli et al. (2017) propose a sequence labeling approach; Subramanian et al. (2018) use pointer networks to select spans from source text; Sun et al. (2019) leverage graph neural networks to help extraction. Despite having huge improvement over tradition approaches, the above methods do not have the capability of predicting absent keyphrases.

Meng et al. (2017) first propose the CopyRNN, a neural model that both generates words from vocabulary and points to words from the source text — overcoming the barrier of predicting absent keyphrases. Following this idea, a host of models have been developed. Chen et al. (2018); Zhao and Zhang (2019) leverage the attention mechanism to help reducing duplication and improving coverage. Ye and Wang (2018) propose a semi-supervised training strategy. Yuan et al. (2020) propose to generate a sequence of keyphrases concatenated with a special token, this enables the model to generate variable number of keyphrases. Chen et al. (2019b); Ye and Wang (2018) propose to leverage extra structure information such as the titles to guide the generation. Chan et al. (2019) propose to leverage Reinforcement Learning (RL) methods. Chen et al. (2019a) retrieve similar documents from training data to help producing more accurate keyphrases. Chen et al. (2020) introduce a hierarchical decoding process and an exclusion mechanism to prevent models from generating duplicate phrases.

8 Conclusion

In this work, we present an empirical study on a set of factors that affect the performance of neural keyphrase generation. We conclude our discussion with the following take-aways:

- One2One excels at in-distribution performance, while One2Seq generalizes better on out-of-distribution data.
- Validation frequency can affect testing scores and thus should be reported.
- Larger beam width leads to better performance but the benefit is diminished past a certain point.
- For One2Seq, target ordering is important in greedy decoding (**PRES-ABS** is a decent choice).

• A larger RNN boosts One2One's performance.
• On absent keyphrase generation, One2One outperforms One2Seq. However, One2Seq with transformer can achieve comparable performance .
• More training data significantly improves TRANSFORMER models' performance (see Appendix E for detailed comparisons).

Based on our exploration so far, we consider some promising future directions of keyphrase generation include: 1) more efficient decoding strategies; 2) methods or metrics that are more robust regarding validation frequency; 3) self-supervised representation learning methods that can better leverage the large but noisy MAGKP data.

References

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards.

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*. 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.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. 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.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Sujatha Das Gollapalli, Xiaoli Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *AAAI*, pages 3180–3187. AAAI Press.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. *29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016*.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. *the Fifteenth Conference on Computational Natural Language Learning*.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobidp. *the 5th International Workshop on Semantic Evaluation*.

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.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1318–1327, Singapore. 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. 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.

D Raj Reddy et al. 1977. Speech understanding systems: A summary of results of the five-year research effort. *Department of Computer Science. Camegie-Mell University, Pittsburgh, PA*, 17.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural models for key phrase extraction and question generation. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 78–88, Melbourne, Australia. Association for Computational Linguistics.

Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR19, page 755764, New York, NY, USA. Association for Computing Machinery.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. *AAAI*.

Minmei Wang, Bo Zhao, and Yihua Huang. 2016. Ptr: Phrase-based topical ranking for automatic keyphrase extraction in scientific publications. *23rd International Conference, ICONIP 2016*.

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*, DL '99, pages 254–255, New York, NY, USA. ACM.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W. Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. In *the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4142–4153, Brussels, Belgium. Association for Computational Linguistics.

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*. 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. 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.

A Statistics of Datasets

In Table 3, we provide statistics of the datasets we use in this work. Among them, all except DUC come from scientific publication in Computer Science domain. In which, KP20K and KRAPIVIN both use keywords provided by the authors as keyphrases. INSPEC, NUS, and SEMEVAL, however, contain the author-provided keywords and additional keyphrases provided by third-party annotators. MAGKP is the most noisy, as part of it is annotated automatically by heuristics and systems.

DUC, diffrent from all above, is a keyphrase dataset based on news articles. Since it represents a rather different distribution from scientific publication datasets, hypothetically, obtaining decent test score on DUC requires extra generalizability.

Dataset	#Train	#Valid	#Test	Mean	%Pre
KP20K	≈514k	2k	≈20k	5.3	63.3%
MAGKP	≈2.7M	–	–	12.9	–
INSPEC	–	–	500	9.6	78.5%
KRAPIVIN	–	–	460	5.2	56.2%
NUS	–	–	211	11.5	51.3%
SEMEVAL	–	–	100	15.7	44.5%
DUC	–	–	308	8.1	97.5%

Table 3: Statistics of various datasets. Mean indicates the average numbers of target phrases, %Pre denotes percentage of present keyphrases.

B Decoding Strategies — Additional Results

In this section, we provide detailed results mentioned in Section 4.2.

- Figure 8 corresponds to Figure 4 (upper left).
- Figure 9 corresponds to Figure 4 (upper right).
- Figure 10 corresponds to Figure 4 (lower left).
- Figure 11 corresponds to Figure 4 (lower right).

C Does Order Matter in One2Seq? — Additional Results

In Section 5, we show a One2Seq model's performance using different target ordering strategies on present keyphrase generation task. We provide the model's performance on absent keyphrase generation task in Table 12.

D Big Data, Big Model? — Additional Results

In Section 6, we show various model variants' average testing performance across different datasets. To provide a clearer view, we provide detailed testing results on each of the datasets here.

In Figure 13 and Figure 14, we show model variants' testing performance when trained with the One2One paradigm.

In Figure 15 and Figure 16, we show model variants' testing performance when trained with the One2Seq paradigm.

E Complete Results

In this section, we report the full set of our experimental results. In Table 4, we report all the testing scores on present keyphrase generation tasks. In addition to the $\mathbf{F}_1$**@10** used throughout this paper, we also report $\mathbf{F}_1$**@5** and $\mathbf{F}_1@\mathcal{O}$ (a metric designed to evaluate systems that capable of generating various number of keyphrases, proposed in (Yuan et al., 2020)) for comparison.

In Table 5, we report all the testing scores on absent keyphrase generation tasks. For absent keyphrase generation, we report **R@10** and **R@50** scores.

F Implementation Details

All our experiments are conducted using the open sourced code provided in https://github.com/memray/OpenNMT-kpg-release.

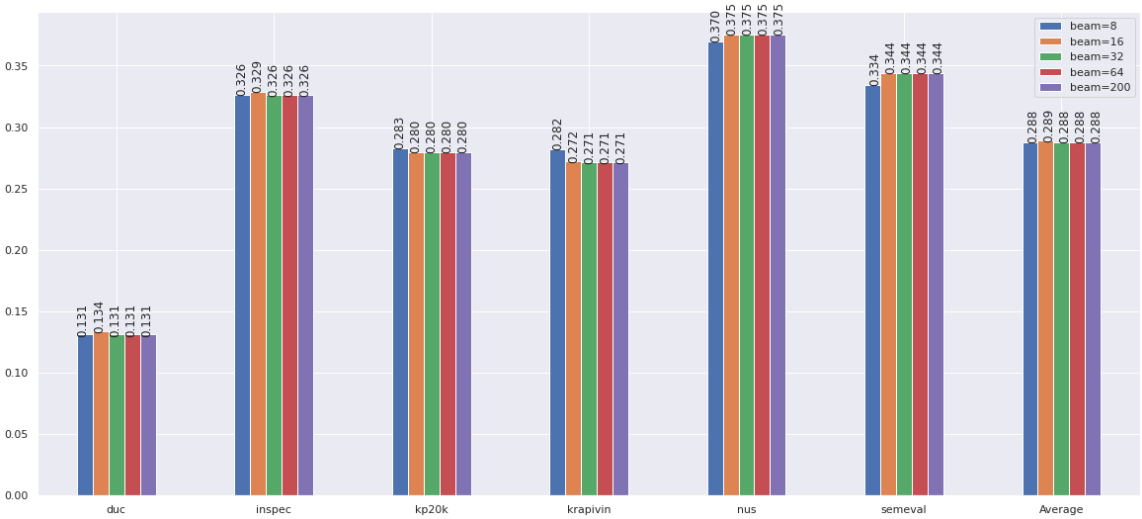

Figure 8: Testing $\mathbf{F}_1$**@10** of a `One2One` model on present keyphrase generation task.

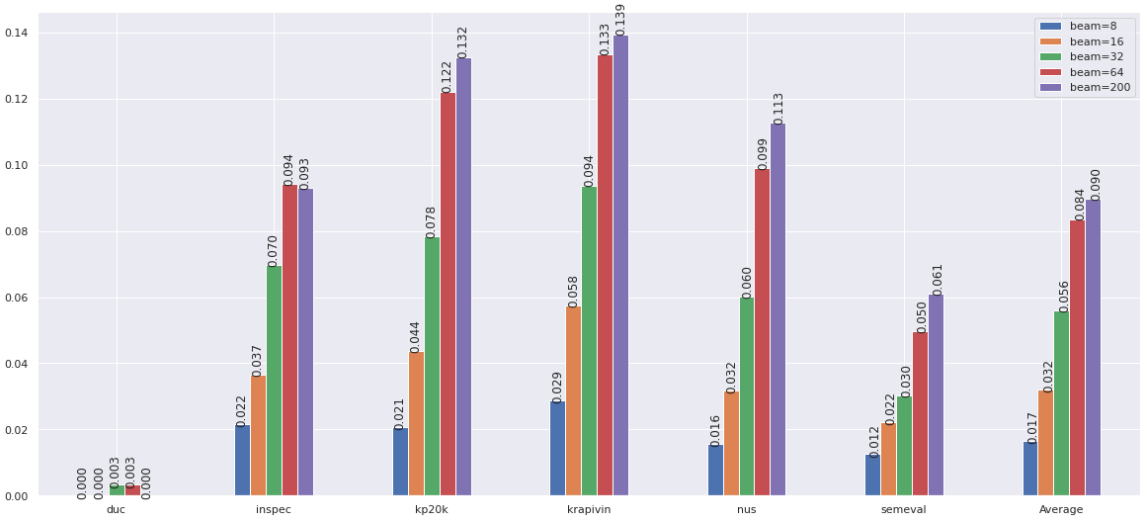

Figure 9: Testing **R@50** of a `One2One` model on absent keyphrase generation task.

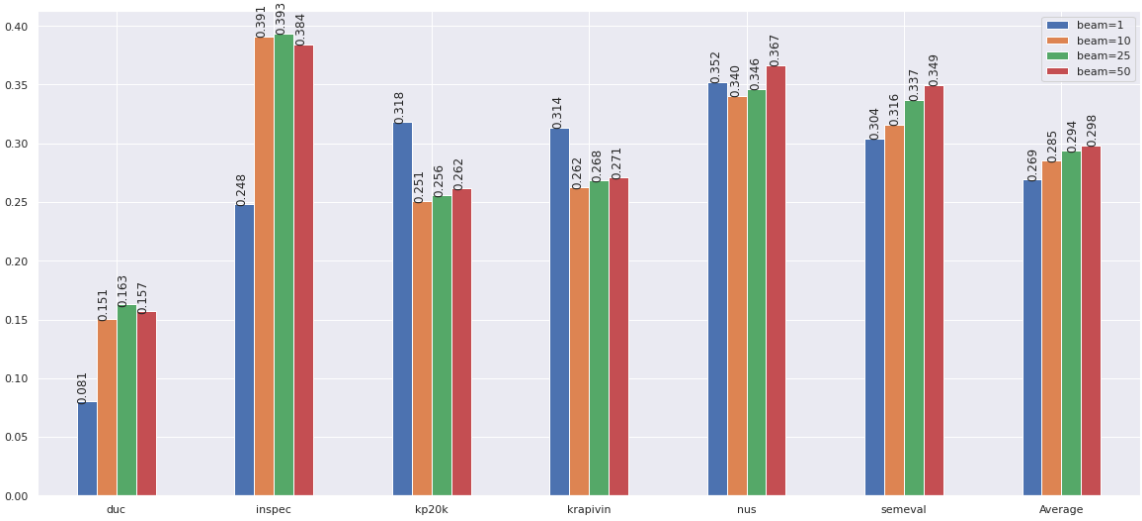

Figure 10: Testing $\mathbf{F}_1$**@10** of a `One2Seq` model on present keyphrase generation task.

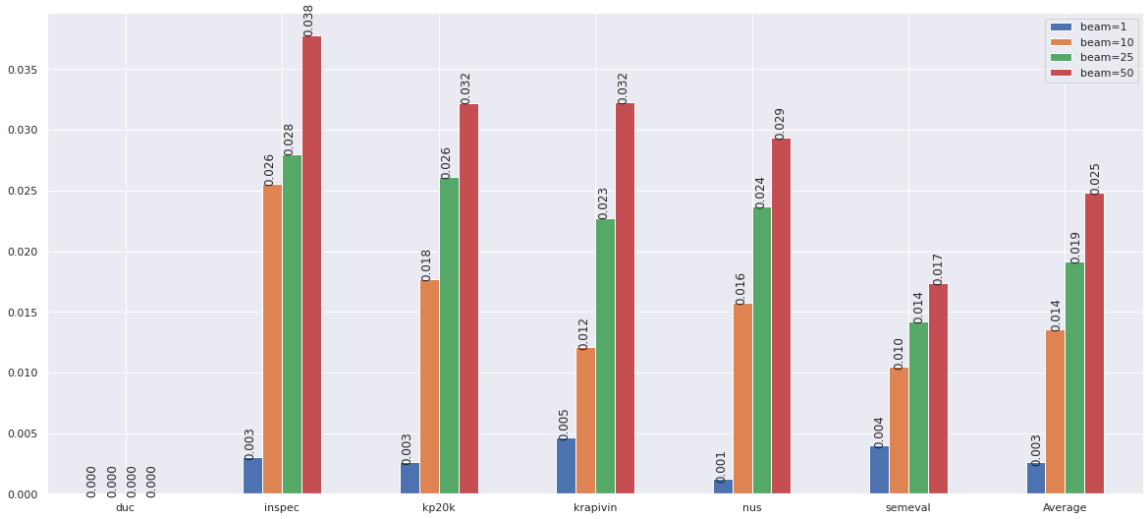

Figure 11: Testing **R@50** of a `One2Seq` model on absent keyphrase generation task.

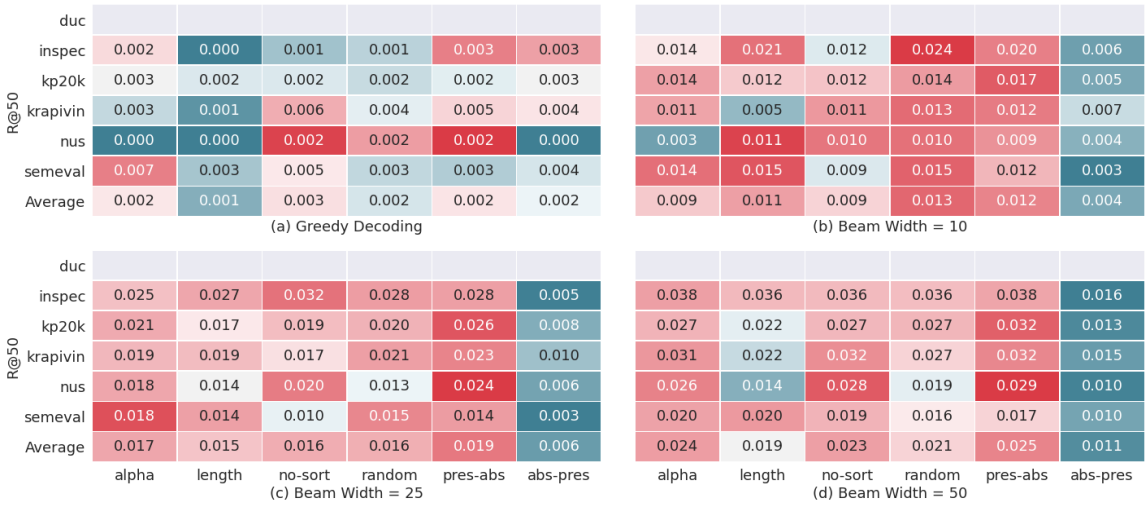

Figure 12: Absent keyphrase generation testing scores on **R@50**. Coloring (from blue to red) represents the relative performance, from low to high, normalized per row.

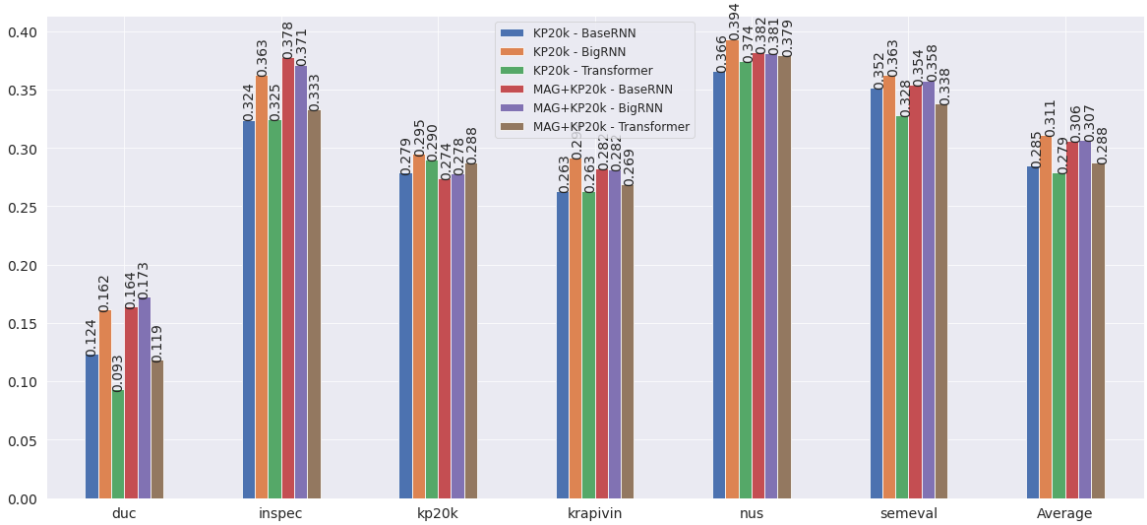

Figure 13: Testing $\mathbf{F}_1$**@10** of different model variants on present keyphrase generation task. Models are trained with `One2One` paradigm.

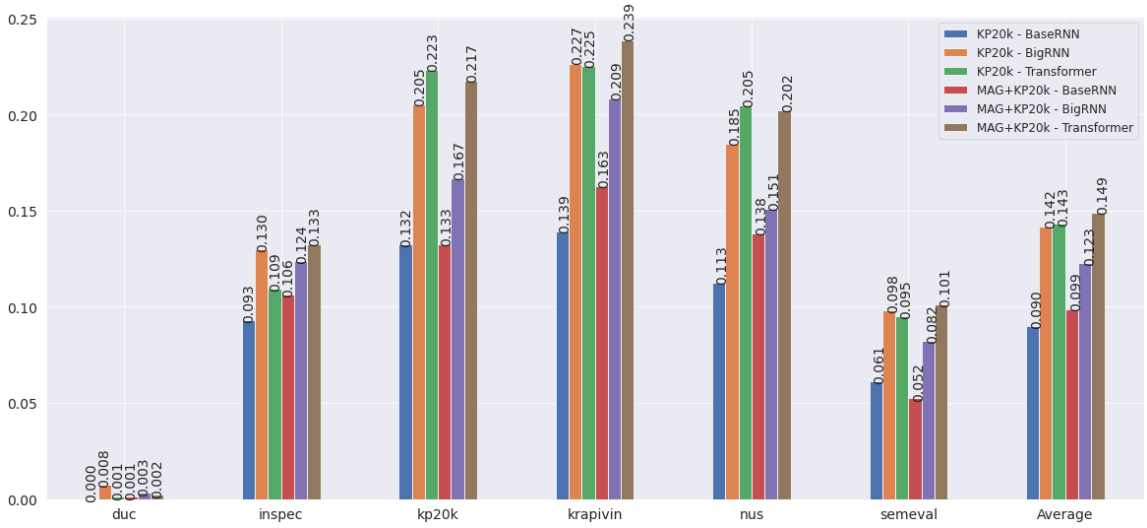

Figure 14: Testing **R@50** of different model variants on absent keyphrase generation task. Models are trained with `One2One` paradigm.

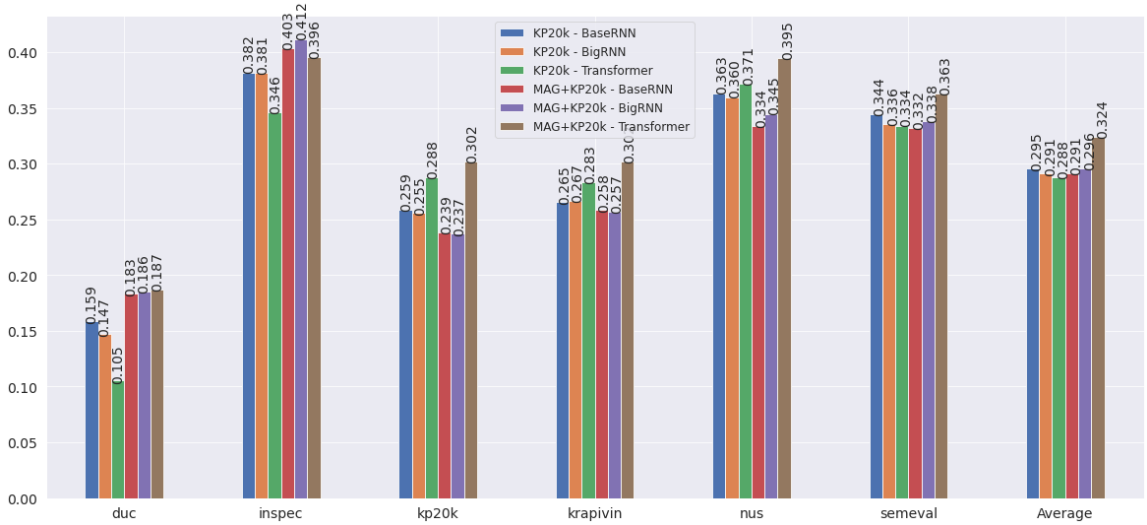

Figure 15: Testing $\mathbf{F}_1$**@10** of different model variants on present keyphrase generation task. Models are trained with `One2Seq` paradigm.

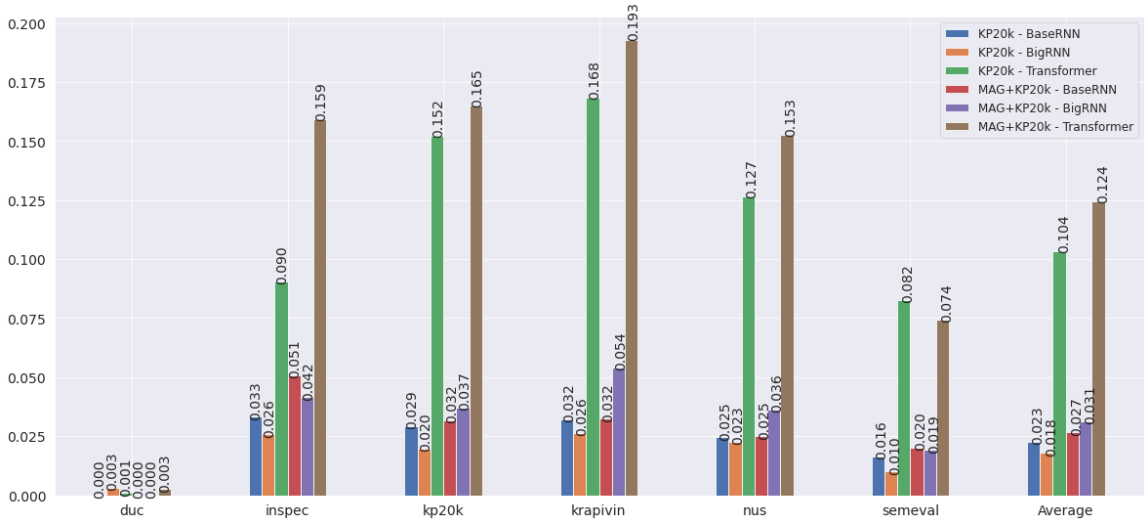

Figure 16: Testing **R@50** of different model variants on absent keyphrase generation task. Models are trained with `One2Seq` paradigm.

Model	Kp20K			Inspec			Krapivin			NUS			SemEval			DUC		
	$F_1@5$	$F_1@10$	$F_1@\mathcal{O}$	$F_1@5$	$F_1@10$	$F_1@\mathcal{O}$	$F_1@5$	$F_1@10$	$F_1@\mathcal{O}$	$F_1@5$	$F_1@10$	$F_1@\mathcal{O}$	$F_1@5$	$F_1@10$	$F_1@\mathcal{O}$	$F_1@5$	$F_1@10$	$F_1@\mathcal{O}$
One2One variants																		
BASERNN	33.1	27.9	35.6	29.2	32.4	33.8	31.8	26.3	34.2	39.2	36.6	43.6	33.5	35.2	35.2	11.2	12.4	11.9
BIGRNN	35.5	29.5	**38.1**	31.1	36.3	37.1	34.2	29.2	**38.8**	**42.5**	39.4	45.7	34.5	**36.3**	36.9	13.5	16.2	15.1
TRANSFORMER	34.7	29.0	37.7	29.2	32.5	33.7	31.3	26.3	36.1	40.7	37.4	44.0	32.3	32.8	35.3	8.3	9.3	9.1
BASERNN +MAGKP	32.4	27.4	34.7	32.3	37.8	38.4	32.4	28.2	35.6	40.2	38.2	43.4	35.4	35.4	37.4	14.2	16.4	16.4
BIGRNN +MAGKP	33.2	27.8	35.3	32.4	37.1	37.9	32.4	28.2	36.5	41.3	38.1	44.8	35.7	35.8	36.0	14.3	17.3	16.2
TRANSFORMER +MAGKP	34.4	28.8	37.0	29.4	33.3	34.5	31.4	26.9	34.1	40.1	37.9	43.0	33.1	33.8	35.6	10.8	11.9	11.0
One2Seq – BASERNN																		
ALPHA	32.8	27.3	33.5	31.3	36.9	37.2	33.2	28.1	35.7	39.2	37.7	42.3	34.8	35.1	37.1	13.6	16.1	15.3
LENGTH	33.1	26.8	33.9	32.3	36.3	36.9	32.7	27.4	34.5	39.1	36.0	41.3	34.3	34.9	36.3	13.4	16.9	15.8
NO-SORT	32.4	26.2	34.4	34.5	39.8	**41.1**	33.1	28.0	36.5	40.0	36.3	42.0	35.5	35.3	36.5	14.0	16.8	15.8
RANDOM	32.2	26.1	34.1	33.9	38.2	38.9	33.0	27.7	35.7	39.8	36.3	43.1	**36.2**	36.2	37.8	14.6	17.1	16.5
PRES-ABS	31.2	26.2	31.0	32.2	38.4	38.4	31.1	27.1	33.2	36.8	36.7	39.2	32.8	34.9	36.3	11.6	15.7	14.5
ABS-PRES	31.8	27.1	32.5	31.5	36.6	37.2	31.0	27.8	34.1	38.4	37.0	40.7	32.7	35.4	35.2	10.6	14.6	12.7
One2Seq variants (trained in PRES-ABS)																		
BASERNN	31.2	26.2	31.0	32.2	38.4	38.4	31.1	27.1	33.2	36.8	36.7	39.2	32.8	34.9	36.3	11.6	15.7	14.5
BIGRNN	30.0	25.5	30.1	32.3	38.1	38.0	30.0	26.7	32.2	37.0	36.0	39.7	33.6	33.6	36.3	11.5	14.7	13.7
TRANSFORMER	34.1	28.8	35.4	30.1	34.6	35.2	32.3	28.3	34.4	39.6	37.1	41.9	32.1	33.4	32.7	9.5	10.5	10.4
BASERNN + MAGKP	28.3	23.9	28.2	32.9	40.3	39.6	28.1	25.8	30.7	35.2	33.4	36.4	30.9	33.2	34.2	13.3	18.3	17.2
BIGRNN + MAGKP	28.2	23.7	28.2	**34.8**	**41.2**	40.1	29.0	25.7	31.0	36.0	34.5	37.8	32.1	33.8	34.8	14.3	18.6	17.4
TRANSFORMER + MAGKP	**36.9**	**30.2**	37.9	33.4	39.6	39.9	**35.1**	**30.2**	35.7	**42.5**	**39.5**	**46.7**	**36.2**	**36.3**	**39.9**	**15.4**	**18.7**	**18.2**
Abstractive Neural Generation																		
CopyRNN (Meng et al.)	32.8	25.5	–	29.2	33.6	–	30.2	25.2	–	34.2	31.7	–	29.1	29.6	–			
CopyRNN* (Yuan et al.)	31.7	27.3	33.5	24.4	28.9	29.0	30.5	26.6	32.5	37.6	35.2	40.6	31.8	31.8	31.7			
CorrRNN (Chen et al.)	-	-	-	-	-	-	31.8	27.8	-	35.8	33.0	-	32.0	32.0	-			
ParaNetT +CoAtt (Zhao and Zhang)	36.0	28.9	-	29.6	35.7	-	32.9	28.2	-	36.0	35.0	-	31.1	31.2	-			
catSeqTG-2RF1† (Chan et al.)	32.1	-	-	25.3	-	-	30.0	-	-	37.5	-	-	28.7	-	-			
KG-KE-KR-M† (Chen et al.)	31.7	28.2	-	25.7	28.4	-	27.2	25.0	-	28.9	28.6	-	20.2	22.3	-			
CatSeq (Yuan et al.)	31.4	27.3	31.9	29.0	30.0	30.7	30.7	27.4	32.4	35.9	34.9	38.3	30.2	30.6	31.0			
CatSeqD (Yuan et al.)	34.8	29.8	35.7	27.6	33.3	33.1	32.5	28.5	37.1	37.4	36.6	40.6	32.7	35.2	35.7			

Table 4: Detailed performance (F_1-score) of present keyphrase prediction on six datasets. **Boldface**/Underline text indicates the best/2nd-best performance in corresponding columns.

	Kp20K		Inspec		Krapivin		NUS		SemEval		DUC	
Model	**R@10**	**R@50**	**R@10**	**R@50**	**R@10**	**R@50**	**R@10**	**R@50**	**R@10**	**R@50**	**R@10**	**R@50**
One2One variants												
BASERNN	6.8	13.2	4.5	9.3	7.9	13.9	4.7	11.3	2.2	6.1	0.0	0.0
BIGRNN	11.6	20.5	5.9	13.0	13.1	22.7	10.4	18.5	4.4	9.8	**0.4**	**0.8**
TRANSFORMER	12.6	**22.3**	5.2	10.9	11.7	22.5	10.7	**20.5**	5.4	9.5	0.0	0.1
BASERNN +MAGKP	6.6	13.3	5.7	10.6	8.4	16.3	7.0	13.8	3.2	5.2	0.0	0.1
BIGRNN +MAGKP	9.1	16.7	7.1	12.4	12.5	20.9	7.6	15.1	3.6	8.2	0.3	0.3
TRANSFORMER +MAGKP	12.5	21.7	6.2	13.3	13.2	**23.9**	11.4	20.2	**6.1**	**10.1**	0.1	0.2
One2Seq - BASERNN												
ALPHA	2.5	2.7	3.4	3.8	2.9	3.1	2.4	2.6	1.8	2.0	0.0	0.0
LENGTH	2.1	2.2	2.8	3.6	2.1	2.2	1.3	1.4	1.9	2.0	0.0	0.0
NO-SORT	2.3	2.7	3.0	3.6	2.7	3.2	2.3	2.8	1.2	1.9	0.0	0.0
RANDOM	2.4	2.7	3.3	3.6	2.4	2.7	1.5	1.9	1.5	1.6	0.0	0.0
PRES-ABS	1.2	1.3	1.6	1.6	1.5	1.5	1.0	1.0	1.0	1.0	0.0	0.0
ABS-PRES	2.7	3.2	3.5	3.8	2.8	3.2	2.6	2.9	1.6	1.7	0.0	0.0
One2Seq variants (trained in PRES-ABS)												
BASERNN	2.7	3.2	3.5	3.8	2.8	3.2	2.6	2.9	1.6	1.7	0.0	0.0
BIGRNN	1.9	2.0	2.5	2.6	2.6	2.6	2.2	2.3	1.0	1.0	0.3	0.3
TRANSFORMER	11.8	15.2	5.9	9.0	13.4	16.8	10.0	12.7	5.7	8.2	0.1	0.1
BASERNN + MAGKP	2.6	3.2	3.7	5.1	2.4	3.2	1.4	2.5	2.0	2.0	0.0	0.0
BIGRNN + MAGKP	3.2	3.7	3.7	4.2	4.6	5.4	2.9	3.6	1.8	1.9	0.0	0.0
TRANSFORMER + MAGKP	**13.8**	16.5	**10.8**	**15.9**	**15.9**	19.3	**12.2**	15.3	5.4	7.4	0.1	0.3

Table 5: Detailed performance (recall scores) of absent keyphrase prediction on six datasets.