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 (KPG) works, including the release of several large-scale datasets and a host of new models to tackle them. Model performance on KPG tasks has increased significantly with evolving deep learning research. However, there lacks a comprehensive comparison among different model designs, and a thorough investigation on related factors that may affect a KPG system's generalization performance. In this empirical study, we aim to fill this gap by providing extensive experimental results and analyzing the most crucial factors impacting the generalizability of KPG models. We hope this study can help clarify some of the uncertainties surrounding the KPG 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. The task can be (and has often been) easily misunderstood and trivialized as yet another natural language generation task like summarization and translation, failing to recognize one key aspect that distinguishes KPG: the multiplicity of generation targets; for each input sequence, a KPG system is expected to output *multiple* keyphrases, each a mini-sequence of multiple word tokens.

Despite this unique nature, KPG has been essentially "brute-forced" into the sequence-to-sequence (Seq2Seq) (Sutskever et al., 2014) framework 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).The community has approached the unique challenges with much ingenuity in problem formulation, model design, and evaluation. For example, multiple target phrases have been reformulated by either splitting into one phrase per data point or joining into a single sequence with delimiters (Figure 1), both allowing straightforward applications of existing neural techniques such as Seq2Seq. In accordance with the tremendous success and demonstrated effectiveness of neural approaches, steady progress has been made in the past few years — at least empirically — across various domains, including sub-areas where it was previously shown to be rather difficult (e.g., in generating keyphrases that are not present in the source text).

Meanwhile, with the myriad of KPG's unique challenges comes an ever-growing collection of studies that, albeit novel and practical, may quickly proliferate and overwhelm. We are therefore motivated to present this study as — to the best of our knowledge — the first systematic investigation on such challenges as well as the effect of interplay among their solutions. We hope this study can serve as a practical guide to help researchers to gain a more holistic view on the task, and to profit from the empirical results of our investigations on a variety of topics in KPG including model design, evaluation, and hyper-parameter selection.

The rest of the paper is organized as follows. We first enumerate specific challenges in KPG due to the multiplicity of its target, and describe general setups for the experiments. We subsequently present experimental results and discussions to answer three main questions:
1. How well do KPG models generalize to various testing distributions?
2. Does the order of target keyphrases matter while training `One2Seq`?
3. Are larger training data helpful? How to better make use of them?

2 Unique Challenges in KPG

Due to the multiplicity of the generation targets, KPG is unique compared to other NLG tasks such

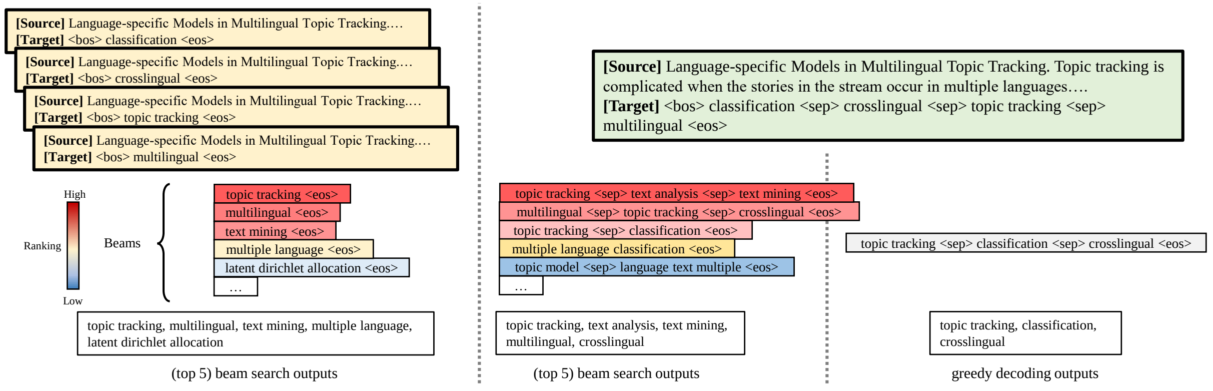

[Source] Language-specific Models in Multilingual Topic Tracking....
[Target] <bos> classification <eos>
[Source] Language-specific Models in Multilingual Topic Tracking....
[Target] <bos> crosslingual <eos>
[Source] Language-specific Models in Multilingual Topic Tracking....
[Target] <bos> topic tracking <eos>
[Source] Language-specific Models in Multilingual Topic Tracking....
[Target] <bos> multilingual <eos>
High
Ranking
Low
Beams
topic tracking <eos>
multilingual <eos>
text mining <eos>
multiple language <eos>
latent dirichlet allocation <eos>
...
topic tracking, multilingual, text mining, multiple language, latent dirichlet allocation
(top 5) beam search outputs
[Source] Language-specific Models in Multilingual Topic Tracking. Topic tracking is complicated when the stories in the stream occur in multiple languages....
[Target] <bos> classification <sep> crosslingual <sep> topic tracking <sep> multilingual <eos>
topic tracking <sep> text analysis <sep> text mining <eos>
multilingual <sep> topic tracking <sep> crosslingual <eos>
topic tracking <sep> classification <eos>
multiple language classification <eos>
topic model <sep> language text multiple <eos>
...
topic tracking, text analysis, text mining, multilingual, crosslingual
(top 5) beam search outputs
topic tracking <sep> classification <sep> crosslingual <eos>
topic tracking, classification, crosslingual
greedy decoding outputs

Figure 1: **Top:** comparison between `One2One` (left) and `One2Seq` (right) paradigms on the same data point. **Bottom:** demonstration of the decoding process for `One2One` (left) and `One2Seq` (mid/right) models. `One2Seq` can apply both beam search (mid) and greedy decoding (right).

as summarization and translation. In this section, we start from providing background knowledge of the KPG problem setup. Then we enumerate the unique aspects in KPG model designing and training that we focus on in this work.

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 sub-string of the source text, or as *absent* otherwise.

Training Paradigms To tackle the unique challenge of generating multiple targets, existing neural KPG approaches 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 framework. 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, `One2Seq` concatenates all ground-truth keyphrases p_i into a single string: $P = \texttt{<bos>}p_1\texttt{<sep>}\cdots\texttt{<sep>}p_n\texttt{<eos>}$ (i.e., prefixed with `<bos>`, joint with `<sep>`, and suffixed with `<eos>`), thus forming a single data point (t, P). A system is then trained to predict the concatenated sequence P given t. By default, we construct P follow the 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 §4.

Architecture In this paper, we adopt the architecture used in both Meng et al. (2017) and Yuan et al. (2020), using `RNN` to denote it. `RNN` 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). We also consider a more recent architecture, Transformer (Vaswani et al., 2017), which is widely used in encoder-decoder language generation literature (Gehrmann et al., 2018). We replace both the encoder GRU and decoder GRU in `RNN` by Transformer blocks, and denote this architecture variant as `TRANS`. Both the `RNN` and `TRANS` models can be trained with either the `One2One` or `One2Seq` paradigm.

In recent years, a host of auxiliary designs and mechanisms have been proposed and developed based on either `One2One` or `One2Seq` (see §6). In this study, however, we focus only on the "vanilla" version of them and we show that given a set of carefully chosen architectures and training strategies, base models can achieve comparable, if not better performance than state-of-the-art methods. We assume that KPG systems derived from either `One2One` or `One2Seq` model would be affected by these factors of model designing in similar ways.

Decoding Strategies KPG is distinct from other NLG tasks since it expects a *set* of multi-word phrases (rather than a single sequence) as model predictions. Depending on the preference of po-

	Dataset	Present ($\mathbf{F_1@\mathcal{O}}$)				Present ($\mathbf{F_1@10}$)				Absent (**R@50**)			
		One2One		One2Seq		One2One		One2Seq		One2One		One2Seq	
		RNN	TRANS	RNN	TRANS	RNN	TRANS	RNN	TRANS	RNN	TRANS	RNN	TRANS
D_0	KP20K	35.5	37.4	31.2	36.2	27.9	28.9	26.1	29.0	13.3	22.3	3.2	15.1
	KRAPIVIN	36.0	33.2	33.5	36.4	26.9	26.5	27.0	28.2	13.9	24.0	3.2	16.8
	D_0 Average	35.7	35.3	32.3	**36.3**	27.4	27.7	26.5	**28.6**	13.6	**23.1**	3.2	15.9
D_1	INSPEC	33.6	32.6	38.6	37.1	32.4	30.9	38.8	36.8	9.3	10.4	3.8	8.1
	NUS	43.3	41.3	39.2	42.2	36.2	36.1	36.6	37.5	11.3	18.9	2.9	12.5
	SEMEVAL	34.8	35.3	36.2	34.8	35.2	33.0	35.2	34.2	6.1	9.9	1.7	8.3
	D_1 Average	37.2	36.4	**38.0**	**38.0**	34.6	33.3	**36.9**	36.1	8.9	**13.1**	2.8	9.6
D_2	DUC	13.2	7.8	**14.7**	11.1	13.9	8.3	**15.9**	11.5	0.0	**0.2**	0.0	0.0
All	Average	32.7	31.3	32.3	**33.0**	28.7	27.3	**29.9**	29.5	9.0	**14.3**	2.5	10.1

Table 1: Testing scores across different model architectures, training paradigms, and datasets. In which, D_0: in-distribution; D_1: out-of-distribution, and D_2: out-of-domain. We provide the average score over each category.

tential downstream tasks, a KPG system can utilize different decoding strategies. For applications that favor *high recall* (e.g., generating indexing terms for retrieval systems), a common practice is to utilize beam search and take predictions from *all* beams[1]. This is applicable in both One2One- and One2Seq-based models to proliferate the number of predicted phrases at inference time. In this work, we use a beam width of 200 and 50 for One2One and One2Seq, respectively. On the contrary, some other applications favor *high precision* and small number of predictions (e.g., KG construction), a One2Seq-based model is capable of decoding greedily, thanks to its nature of generating multiple keyphrases in a sequential manner.

As an example, we illustrate the two decoding strategies in Figure 1. Specifically, a One2One model typically collects output keyphrases from all beams and use the top k phrases as the model output ($k = 5$ in the example). In One2Seq, either beam search or greedy decoding can be applied. For beam search, we use both the order of phrases within a beam and the rankings of beams to rank the outputs. In the shown example, top 5 beam search outputs are obtained from the 2 beams with highest rankings. As for greedy decoding, the decoder uses a beam size of 1, and takes all phrases from the single beam as outputs. In this way, the One2Seq model can determine the number of phrases to output by itself conditioned on t.

Evaluation Due to the multiplicity of targets in KPG task, the evaluation protocols are distinct from typical NLG tasks. A spectrum of evaluation metrics have been used to evaluate KPG systems, including metrics that truncate model outputs at a fixed number such as $\mathbf{F_1@5}$ and $\mathbf{F_1@10}$ (Meng et al., 2017); metrics that evaluate a model's ability of generating variant number of phrases such as $\mathbf{F_1@\mathcal{O}}$ and $\mathbf{F_1@\mathcal{M}}$ (Yuan et al., 2020); metrics that evaluate absent keyphrases such as Recall@50 (**R@50**). Detailed definitions of the metrics are provided in Appendix A. Due to space limit, we mainly discuss $\mathbf{F_1@\mathcal{O}}$, $\mathbf{F_1@10}$ and **R@50** in the main content, complete results with all common metrics are included in Appendix E. We save model checkpoints for every 5,000 training steps and report test performance using checkpoints that produce the best $\mathbf{F_1@\mathcal{O}}$ or **R@50** on the KP20K validation set.

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 KPG 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 datasets without fine-tuning. Details of the datasets are shown in Appendix B.

3 Generalizability

In this section, we show and analyze the generalization performance of KPG systems from 2 dimensions: model architecture and training paradigm. Specifically, we compare the two model architectures (i.e., RNN and TRANS) as described in §2. For each model architecture, we train the KPG model using either of the training paradigms (i.e., One2One or One2Seq) also as described in §2.

To better understand model variants' generalization properties, we categorize the 6 testing sets into 3 classes according to their distribution similarity with the training data (KP20K), as shown in Table 1. Concretely, KP20K and KRAPIVIN are in-distribution test sets (denoted as D_0), since they both contain scientific paper abstracts paired with keyphrases provided by their authors. INSPEC, NUS and SEMEVAL are out-of-distribution test sets

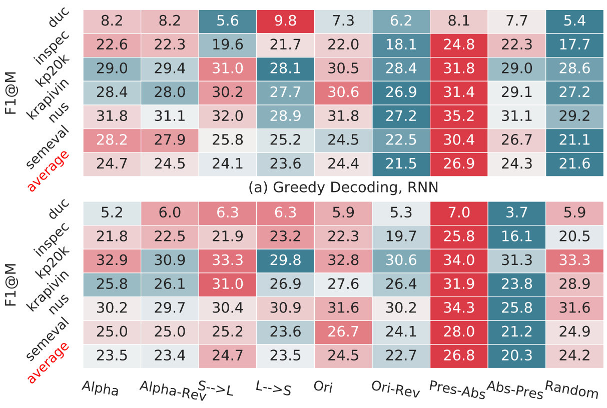
[1]This is in contrast to only taking the *single* top beam as in typical NLG tasks.

(denoted as D_1), they share same type of source text with D_0, but with additionally labeled keywords by third-party annotators. DUC is a special test set which uses news articles as its source text. Because it shares the least domain knowledge and vocabulary with all the other test sets, we call it out-of-domain test set (denoted as D_2).

Model Architecture: RNN vs TRANS The first thing to notice is that on present KPG, the models show consistent trends between $\mathbf{F_1@10}$ and $\mathbf{F_1@\mathcal{O}}$. We observe that TRANS models significantly outperform RNN models when trained with the One2Seq paradigm on D_0 test sets. However, when test data distribution shift increases, on D_1 test sets, RNN models starts to outperform TRANS; eventually, when dealing with D_2 test set, RNN outperforms TRANS by a large margin. On models trained with One2One paradigm, we observe a similar trend. On D_0 data, TRANS models achieve comparable $\mathbf{F_1@10}$ and $\mathbf{F_1@\mathcal{O}}$ scores with RNN, when data distribution shift increases, RNN models produce better results.

On the contrary, for absent KPG, TRANS outperforms RNN by a significant margin in all experiment settings. This is especially obvious when models are trained with One2Seq paradigm, where RNN models barely generalize to any of the testing data and produce an average **R@50** of 2.5. In the same setting, TRANS models get an average **R@50** of 10.1, which is $4\times$ higher than RNN.

To further study the different generation behaviors between RNN and TRANS, we investigate the average number of unique predictions generated by either of the models. As shown in Figure 12 in Appendix D, comparing results of order **PRES-ABS** in sub-figure a/b (RNN) with sub-figure c/d (TRANS), we observe that TRANS is consistently generating more unique predictions than RNN, in both cases of greedy decoding (4.5 vs 4.2) and beam search (104.3 vs 97.7). We suspect that generating a more diverse set of keyphrases may have a stronger effect on in-distribution test data. The generated outputs during inference are likely to represent the distribution learned from the training data, when the test data share the same (or similar) distribution, a larger set of unique predictions leads to a higher recall — which further contributes to their F-scores. In contrast, on test sets which data distribution is far from training distribution, the extra predictions may not be as useful, and even hurts precision. Similarly, because we evaluate absent KPG by the models' recall, TRANS models — produce more unique predictions — can always outperform RNN models.[2]



(a) Greedy Decoding, RNN

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	8.2	8.2	5.6	9.8	7.3	6.2	8.1	7.7	5.4
inspec	22.6	22.3	19.6	21.7	22.0	18.1	24.8	22.3	17.7
kp20k	29.0	29.4	31.0	28.1	30.5	28.4	31.8	29.0	28.6
krapivin	28.4	28.0	30.2	27.7	30.6	26.9	31.4	29.1	27.2
nus	31.8	31.1	32.0	28.9	31.8	27.2	35.2	31.1	29.2
semeval	28.2	27.9	25.8	25.2	24.5	22.5	30.4	26.7	21.1
average	24.7	24.5	24.1	23.6	24.4	21.5	26.9	24.3	21.6

(b) Greedy Decoding, Transformer

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	5.2	6.0	6.3	6.3	5.9	5.3	7.0	3.7	5.9
inspec	21.8	22.5	21.9	23.2	22.3	19.7	25.8	16.1	20.5
kp20k	32.9	30.9	33.3	29.8	32.8	30.6	34.0	31.3	33.3
krapivin	25.8	26.1	31.0	26.9	27.6	26.4	31.9	23.8	28.9
nus	30.2	29.7	30.4	30.9	31.6	30.2	34.3	25.8	31.6
semeval	25.0	25.0	25.2	23.6	26.7	24.1	28.0	21.2	24.9
average	23.5	23.4	24.7	23.5	24.5	22.7	26.8	20.3	24.2

Figure 2: Present KPG testing scores ($\mathbf{F_1@\mathcal{M}}$). Colors represent the relative performance, normalized per row.

Training Paradigm: One2One vs One2Seq We observe that on present KPG tasks, models trained with the One2Seq paradigm outperforms One2One in most settings, this is particularly clear on D_1 and D_2 test sets. We believe this is potentially due to the unique design of the One2Seq training paradigm where at every generation step, the model conditions its decision making on all previously generated tokens (phrases). Compared to the One2One paradigm where multiple phrases can only be generated independently by beam search in parallel, the One2Seq paradigm can model the dependencies among tokens and the dependencies among phrases more explicitly.

However, on absent KPG, One2One consistently outperforms One2Seq. Furthermore, only when trained with One2One paradigm, an RNN-based model can achieve **R@50** scores close to TRANS-based models. This may because a One2Seq model tends to produce more duplicated predictions during beam search inference. By design, every beam is a string that contains multiple phrases that concatenated by the delimiter <sep>, there is no guarantee that the phrase will not appear in multiple beams. In the example shown in Figure 1, "topic tracking" is such a duplicate prediction that appears in multiple beams. In fact, the proportion of duplicates in One2Seq predictions

[2]Our TRANS and RNN models follow Vaswani et al. (2017) and Meng et al. (2017)'s hyper-parameter settings respectively. RNN is significantly lighter than TRANS. We conduct experiments with a much larger RNN but only observe marginal performance boost against Meng et al. (2017)'s setting.

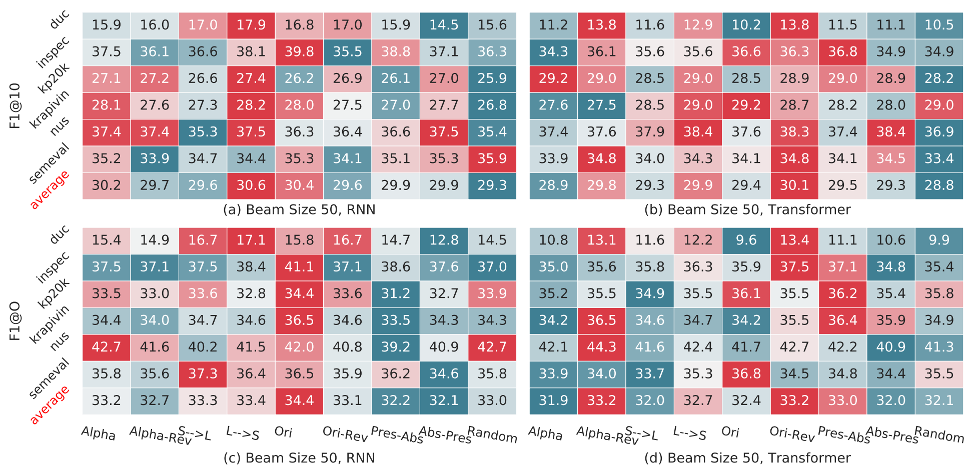


Figure 3: Present KPG testing scores (**top: $\mathbf{F}_1@10$**, **bottom: $\mathbf{F}_1@\mathcal{O}$**). Colors represent the relative performance, normalized per row.

is more than 90%. This is in contrast with beam search on `One2One` models, where each beam only contains a single keyphrase thus has a much lower probability of generating duplication.[3]

4 Does Order Matter in One2Seq?

In the `One2One` paradigm (as shown in Figure 1), 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 has not been 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.

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

- **RANDOM**: Randomly shuffle the target phrases.

Because of the *set generation* nature of KPG, we expect randomly shuffled target sequences help to learn an order-invariant decoder.

- **ORI**: 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).
- **ORI-REV**: Reversed order of **ORI**.
- **S->L**: Phrases sorted by lengths (number of tokens, from short to long).
- **L->S**: Reversed order of **S->L**.
- **ALPHA**: Sort phrases by alphabetical order.
- **ALPHA-REV**: Reversed order of **ALPHA**.
- **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.

Greedy Decoding In Figure 2, we show the `RNN` and `TRANS` model's $\mathbf{F}_1@\mathcal{M}$ on present KPG task, equipped with greedy decoding. In this setting, the model simply chooses the token with the highest probability at every step, and terminates either upon generating the `<eos>` token or reaching the maximum target length limit (40). 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

[3]Due to post-processing such as stemming, `One2One` model may still produce duplication.

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 when greedy decoding is applied. When using the RNN architecture, RANDOM consistently yields lower $\mathbf{F_1@\mathcal{M}}$ than other ordering strategies on all datasets. This suggests that a consistent order of the keyphrases is beneficial. However, TRANS models show a better resistance against randomly shuffled keyphrases and produce average tier performance with the RANDOM ordering. Meanwhile, we observe that 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. We observe similar trends in greedy decoding models' $\mathbf{F_1@\mathcal{O}}$ and $\mathbf{F_1@10}$, due to space limit, we refer readers to Figure 9, 10 in Appendix D.

Beam Search Next, we show results obtained from the same set of models equipped with beam search (beam width is 50) in Figure 3 (a/b). Compared with greedy decoding (Figure 10, Appendix 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). We observe that compared to the greedy decoding case, the pattern among different ordering strategies appears to be less clear, with the scores distributed more evenly across different settings (concretely, the absolute difference between max average score and min average score is lower).

We suspect that the uniformity among different ordering strategies with beam search may be due to the limitation of the evaluation metric $\mathbf{F_1@10}$. The metric $\mathbf{F_1@10}$ truncates a model's predictions to 10 top-ranked keyphrases. By investigation, we find that during greedy decoding, the number of predictions acts as a dominant factor, this number varies greatly among different ordering. With greedy decoding, PRES-ABS can generally predict more phrases than the others, which explains its performance advantage (Figure 13 (a/c), Appendix D). However, as the beam width increases, all models can predict more than 10 phrases (Figure 13 (b/d), Appendix D). In this case, the $\mathbf{F_1@10}$ is contributed more by a model' ability of generating more high quality keyphrases within its top-10 outputs, rather than the amount of predictions. Therefore, the performance gap among ordering strategies is gradually narrowed in beam search. For instance, we observe that the $\mathbf{F_1@10}$ difference between PRES-ABS and S->L produced by RNN is 3.2/2.0/1.0/0.3 when beam width is 1/10/25/50.

To validate our assumption, we further investigate the same set of models' performance on $\mathbf{F_1@\mathcal{O}}$, which strictly truncates the generated keyphrase list by the number of ground-truth keyphrases $\mathcal{O}$ (where in most cases $\mathcal{O} < 10$). Under this harsher criterion, a model is required to generate more high quality keyphrases within its top-$\mathcal{O}$ outputs. From Figure 3 (c/d), we observe that the scores are less uniformly distributed, this indicates a larger difference between different order settings. Among all orders, ORI produces best average $\mathbf{F_1@\mathcal{O}}$ with RNN, whereas ALPHA-REV and ORI-REV produce best average $\mathbf{F_1@\mathcal{O}}$ with TRANS.

In our curated list of order settings, there are 3 pairs of orderings with reversed relationship (i.e., S->L vs L->S, ALPHA vs ALPHA-REV, ORI vs ORI-REV). Interestingly, we observe that when beam search is applied, these orderings often show a non-negligible score difference with their counterparts. This also suggests that order matters since specific model architecture and training paradigm often has its own preference on the phrase ordering.

It is also worth mentioning that when we manually check the output sequences in test set produced by ALPHA ordering, 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.

Ordering in Absent KPG We report the performance of the same set of models on the absent portion of data in Figure 11, Appendix D. Although achieving relatively low $\mathbf{R@50}$ in most settings, scores produced by various orderings show clear distinctions, normalized heat maps suggest that the rankings among different orderings tend to be consistent across all testing datasets. In general, PRES-ABS produces better absent keyphrases across different model architectures. Due to the

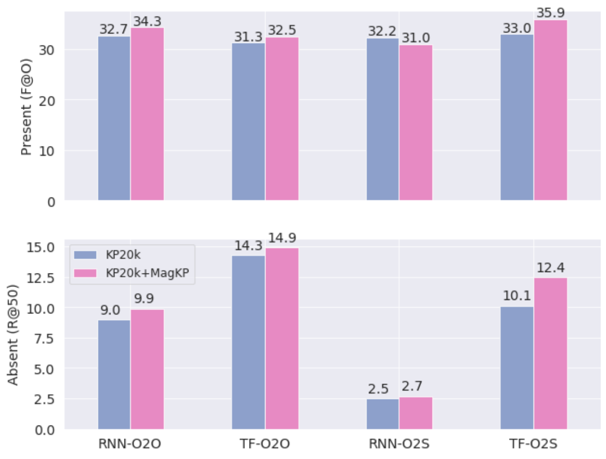


Figure 4: Comparing models trained solely with KP20K against with additional MAGKP data.

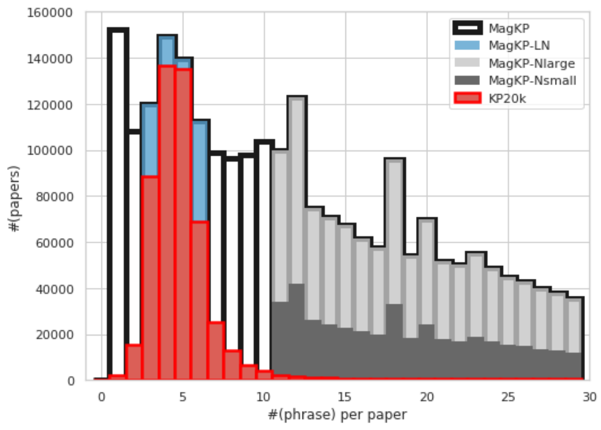


Figure 5: A histogram showing the distribution of #(kp per document) on KP20K, MAGKP and its subsets. Data points with more than 30 keyphrases are truncated.

space limit, we encourage readers to check out Appendix D, which provides an exhaustive set of heat maps including all experiment settings and metrics discussed in this section.

5 Training with More Data

In this section, we further explore the possibility of improving KPG performance by scaling up the training data. Data size has been shown as one of the most effective factors for training language models (Raffel et al., 2019; Ott et al., 2018) but it has yet been discussed in the context of KPG.

MagKP Dataset We construct a new dataset, namely MAGKP, on the basis of Microsoft Academic Graph (Sinha et al., 2015). We filter the original MAG v1 dataset (166 million papers, multiple domains) and only keep papers in *Computer Science* and with at least one keyphrase. This results in 2.7 million data points ($5\times$ larger than KP20K). This dataset remains noisy despite the stringent filtering criteria, this is because 1) the data is crawled from the web and 2) some keywords are labeled by automatic systems rather than humans. This noisy nature brings many interesting observations.

General Observations The first thing we try is to train a KPG model with both KP20K and MAGKP. During training, the two dataset are fed to the model in an alternate manner, we denote this data mixing strategy as ALT. In Figure 4, we compare models' performance when trained on both KP20K and MAGKP against solely on KP20K. We observe the extra MAGKP data brings consistent improvement across most model architecture and training paradigm variants. This suggests that model KPG models discussed in this work can benefit from additional training data. Among all the settings, $\mathbf{F_1@\mathcal{O}}$ of the TRANS+One2Seq is boosted by near 3 points on present KPG, the resulting score outperforms other variants by a significant margin and even surpass a host of state-of-the-art models (see comparison in Appendix E). Again, the same setting obtains a 2.3 boost of **R@50** score on the absent KPG task, makes TRANS+One2Seq the setting that benefits the most from extra data. In contrast, the extra MAGKP data provide only marginal improvement to RNN-based models. On present KPG, RNN+One2Seq even has an $\mathbf{F_1@\mathcal{O}}$ drop when trained with more data.

As mentioned in §3, the RNN model is significantly lighter than TRANS. To investigate if an RNN with more parameters can benefit more from MAGKP, we conduct experiments which use a GRU with much larger hidden size (dubbed BIGRNN). Results (in Appendix E) suggest otherwise, extra training data leads to negative effect on One2One and only marginal gain on One2Seq. We thus believe the architecture difference between TRANS and RNN is the potential cause, for instance, the built-in self-attention mechanism may help TRANS models learning from noisy data.

Learning with Noisy Data To further investigate the performance boost brought by the MAGKP dataset on TRANS+One2Seq, we are curious to know which portion of the noisy data helped the most. As a naturally way to cluster the MAGKP data, we define the noisiness by the number of keyphrases per data point. As shown in Figure 5, the distribution of MAGKP (black border) covers a

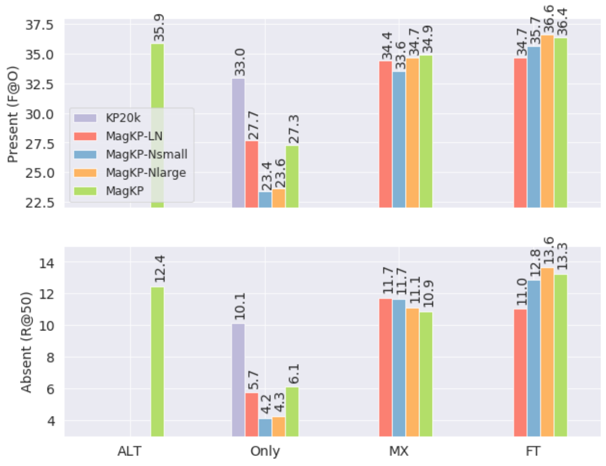


Figure 6: TRANS+One2Seq trained with KP20K and different subsets of MAGKP, using four data mixing strategy. Scores are averaged over all 6 test sets.

much wider spectrum on the x-axis compared to KP20K (red). Because keyphrase labels are provided by human authors, a majority of its keyphrase numbers lie in the range of [3, 6]; however, only less than 20% of the MAGKP data overlaps with this number distribution.

We thus break MAGKP down into a set of smaller subset: 1) MAGKP-LN is a considerably **L**ess **N**oisy subset that contains data points that have 3~6 phrases. 2) MAGKP-Nlarge is the **N**oisy subset in which all data points have more than 10 keyphrases. 3) MAGKP-Nsmall is a randomly sampled subset of MAGKP-Nlarge with the same size as MAGKP-LN.

We also define a set of data mixing strategies to compare against ALT: ONLY: models are trained solely on a single set (or subset) of data; MX: KP20K and MAGKP (or its subset) are split into shards (10k each) and they are randomly sampled during training; FT: models are pre-trained on MAGKP (or its subset) and fine-tuned on KP20K.

In Figure 6, we observe that none of the MAGKP subsets can match KP20K's performance in the ONLY setting. Because MAGKP-LN and MAGKP-Nsmall share similar data size with KP20K, this suggest the distributional shift between MAGKP and the 6 testing sets is significant. In the MX setting where KP20K is mixed with noisy data, we observe a notable performance boost compared to ONLY (yet still lower than ALT), however, we do not see clear patterns among the 4 MAGKP subsets in this setting. In the FT setting, we observe a surge in scores across all MAGKP subsets. In present KPG, both MAGKP and MAGKP-Nlarge outperform the score achieved in the ALT setting; similarly, in absent KPG, MAGKP, MAGKP-Nlarge and MAGKP-Nsmall exceeds the ALT score. This is to our surprise that the subsets considered as noisy provide a greater performance boost, while they perform poorly if "ONLY" trained on these subsets.

To sum up, during our investigation on augmenting KP20K with the noisy MAGKP data, we obtain the best performance from a TRANS+One2Seq model that pre-trained on MAGKP and then fine-tuned on KP20K, and this performance has outperformed current state-or-the-art models. We conjecture that the performance gain may come from data diversity, because MAGKP contains a much wider distribution of data compared to the author keyword distribution as in KP20K. This inspires us to develop data augmentation techniques to exploit the diversity in unlabeled data.

6 Related Work

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 heuristically selects a set of candidate phrases from the text using some pre-defined features (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. The ranker can be bagged decision trees (Medelyan et al., 2009; Lopez and Romary, 2010), Multi-Layer Perceptron, Support Vector Machine (Lopez and Romary, 2010) or 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.

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

Meng et al. (2017) first propose the CopyRNN model, which both generates words from vocabulary and points to words from the source text — overcoming the barrier of predicting absent keyphrases. Following this idea, 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 `One2Seq`, which enables a model to generate variable number of keyphrases. Chen et al. (2019b); Ye and Wang (2018); Wang et al. (2019) propose to leverage extra structure information (e.g., title, topic) to guide the generation. Chan et al. (2019) propose an RL model, Swaminathan et al. (2020) propose using GAN for KPG. Chen et al. (2019a) retrieve similar documents from training data to help producing more accurate keyphrases. Chen et al. (2020) introduce hierarchical decoding and exclusion mechanism to prevent from generating duplication. Çano and Bojar (2019) also propose to utilize more data, but their goal is to bridge KPG with summarization.

7 Conclusion and Takeaways

We present an empirical study discussing neural KPG models from various aspects. Through extensive experiments and analysis, we answer the three questions (§1). Results suggest that given a carefully chosen architecture and training strategy, a base model can perform comparable with fancy SOTA models. Further augmented with (noisy) data in the correct way, a base model can outperform SOTA models (Appendix E). We strive to provide a guideline on how to choose such architectures and training strategies, which hopefully can be proven valuable and helpful to the community.

We conclude our discussion with the following takeaways:

1. `One2Seq` excels at present KPG, while `One2One` performs better on absent KPG. See Section 3.

2. For present KPG, `TRANS` performs better on in-distribution data, when distribution or domain shift increase, `RNN` can outperform `TRANS`. See Section 3.

3. On absent KPG, `TRANS` is the clear winner. See Section 3.

4. For `One2Seq`, target ordering is important in greedy decoding (with **PRES-ABS** being an overall good choice). See Section 4.

5. The effect of target ordering tends to diminish when beam search is performed. See Section 4.

6. Large and noisy data can benefit KPG. Empirically, a decent way to leverage them is to pre-train on extra data then fine-tune on small in-domain data. See Section 5.

7. Copy mechanism helps present prediction while worsening absent performance. See Appendix C.1.

8. Larger beam width is beneficial, especially for absent KPG. However, on present KPG tasks, the benefit is diminished past a certain point and thus computational efficiency needs to be carefully considered. See Appendix C.2.

Acknowledgments

RM is partially supported by the Amazon Research Awards for the project "Transferable, Controllable, Applicable Keyphrase Generation". This research is partially supported by the University of Pittsburgh Center for Research Computing through the resources provided. The authors thank the anonymous NAACL reviewers for their helpful feedback and suggestions.

References

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

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.

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.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

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.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

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*, SIGIR'19, page 755–764, 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.

Avinash Swaminathan, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, Rajiv 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.

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

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*.

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*, 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.

Contents in Appendices:

- In Appendix A, we provide the formal definition of all evaluation metrics we used in this work.
- In Appendix B, we provide detailed statistics of all datasets used in this work.
- In Appendix C, we provide observation and analysis on additional factors that can affect a KPG system's performance.
- In Appendix D, show the set of heat maps that are not shown in the main content due to space limit.
- In Appendix E, we provide a complete set of numbers containing all results discussed in this work, compared with a set of SOTA models in existing literature.
- In Appendix F, we provide implementation details that helps to reproduce our experiments.

A Evaluation Metric Definition

In this section, we provide definition of the metrics we use in this work. All metrics are adopted from (Meng et al., 2017) and (Yuan et al., 2020). To make the results easy to reproduce, we simply *report macro-average scores over all the data examples* in a dataset (rather than removing examples that contain no present/absent phrases). Since some data examples contain no valid present/absent phrase and lead to zero scores, this causes *our results can be lower than previously reported results.*

Given a data example consisting a source text $\mathcal{X}$ and a list of target keyphrases $\mathcal{Y}$, suppose that a model predicts a list of unique keyphrases $\hat{\mathcal{Y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ ordered by the quality of the predictions $\hat{y}_i$, and that the ground truth keyphrases for the given source text is the oracle set $\mathcal{Y}$. When only the top k predictions $\hat{\mathcal{Y}}_{:k} = (\hat{y}_1, \ldots, \hat{y}_{\min(k,m)})$ are used for evaluation, *precision*, *recall*, and F_1-score are consequently conditioned on k and defined as:

$$\mathbf{P}@k = \frac{|\hat{\mathcal{Y}}_{:k} \cap \mathcal{Y}|}{|\hat{\mathcal{Y}}_{:k}|}, \qquad \mathbf{R}@k = \frac{|\hat{\mathcal{Y}}_{:k} \cap \mathcal{Y}|}{|\mathcal{Y}|},$$
$$\mathbf{F}_1@k = \frac{2 * \mathbf{P}@k * \mathbf{R}@k}{\mathbf{P}@k + \mathbf{R}@k}. \tag{1}$$

Thus the metrics are defined as:

- **F$_1$@5**: $\mathbf{F}_1@k$ when $k = 5$.
- **F$_1$@10**: $\mathbf{F}_1@k$ when $k = 10$.
- **F$_1$@$\mathcal{O}$**: $\mathcal{O}$ denotes the number of oracle (ground truth) keyphrases. In this case, $k = |\mathcal{Y}|$, which means for each data example, the number of predicted phrases taken for evaluation is the same as the number of ground-truth keyphrases.
- **F$_1$@$\mathcal{M}$**: $\mathcal{M}$ denotes the number of predicted keyphrases. In this case, $k = |\hat{\mathcal{Y}}|$ and we simply take all the predicted phrases for evaluation without truncation.
- **R@50**: $\mathbf{R}@k$ when $k = 50$.

B Statistics of Datasets

We provide details of datasets used in this work. We use KP20K-train (Meng et al., 2017) and MAGKP (Sinha et al., 2015) for training keyphrase generation models, both are built on the basis of scientific publications in Computer Science domain. Nevertheless, their distributions are considerably different, e.g. MAGKP data contains 3 times more keyphrases on average than KP20K. This is because KP20K is constructed using real author keywords whereas MAGKP may contain a vast amount of keyphrases annotated by automatic systems. Detailed statistics are listed in Table 2. We also leave out certain amount of data points from KP20K for validation and testing (KP20K-VALID and KP20K-TEST).

We also utilize five other datasets for evaluation purposes, as shown in Table 3. All except DUC come from scientific publications in Computer Science domain. KRAPIVIN uses keywords provided by the authors as targets, which is the same as KP20K. INSPEC, NUS, and SEMEVAL contain author-assigned keywords and additional keyphrases provided by third-party annotators. DUC, different 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.

C Other Model Designing Aspects

Besides the findings we discuss in the paper, there exist other important factors affect the general performance of KPG models. We provides two addi-

Dataset	#Data	#kp	#unique kp	#(word) per kp
KP20K-train	514K	2.7M	700K	1.92
MAGKP	2.7M	41.6M	6.9M	3.42
MAGKP-LN	522K	2.3M	579K	2.73
MAGKP-Nlarge	1.5M	35.5M	5.8M	3.38
MAGKP-Nsmall	522K	12.2M	2.2M	3.37

Table 2: Statistics of training datasets.

Dataset	#Data	Avg#kp	%Pre
KP20K	≈20K	5.3	63.3%
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 testing datasets. Avg#kp indicates the average numbers of target keyphrases, %Pre denotes percentage of present keyphrases.

tional empirical results that we think might be of interest to certain readers.

C.1 Effect of Copy Mechanism

Copy mechanism (Gu et al., 2016) (also referred to as Pointer Generator (See et al., 2017) or Pointer Softmax (Gulcehre et al., 2016)) has demonstrated to play a critical role in tasks where texts on the source and target side may overlap, such as summarization (See et al., 2017) and keyphrase generation (Meng et al., 2017). Basically, it is an additional loss that enables models to extract information from the source side with the help of attentions. Prior studies (Meng et al., 2017) have shown the importance of copy mechanism with RNN+One2One, but no further comparison has been made.

In Figure 7, we present the results of four KPG model variants, equipped with and without copy mechanism. The results show that copy mechanism leads to considerable improvements on present KPG, especially for RNN. TRANS benefits less from the copy, which may be because its multi-head attentions behave similarly to the copy mechanism even without explicit training losses. With regard to the absent KPG results, copy mechanism only helps RNN+One2One. This suggests that TRANS can achieve consistently better abstractiveness (absent performance) by disabling the copy mechanism at the cost of weaker extractiveness. This dilemma cautions researchers to use copy mechanism more wisely according to specific applications.

C.2 Effect of Beam Width

As discussed in §2, one unique challenge of the KPG task is due to its multiplicity of target outputs. As a result, a common strategy is to take multiple beams during decoding in order to obtain more phrases (as opposed to greedy decoding). This choice is at times not only practical but in fact necessary: under the One2One paradigm, for example, it is crucial to have multiple beams in order to generate multiple keyphrases for a given input.

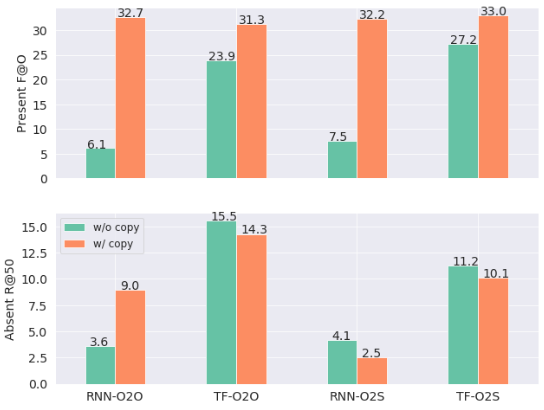


Figure 7: Averaged scores of models with and without utilizing copy mechanism.

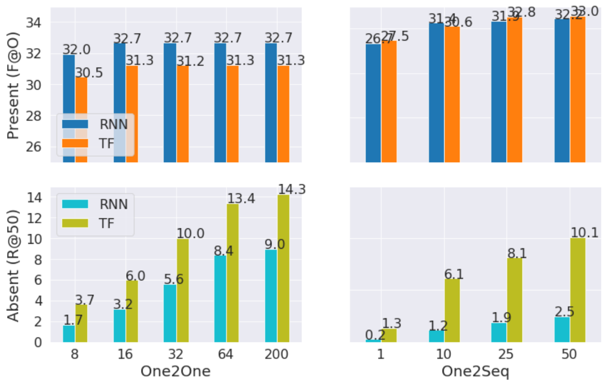


Figure 8: Averaged scores of models using different widths of Beam Search.

Generally speaking, KPG and its evaluation metrics are in general 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, as shown in Figure 8, we observe that beam search can sometimes achieve a relative gain of more than 10% in present phrase generation , and a much larger performance boost in absent phrase generation, over greedy decoding.

We are also interested in seeing if there exists an optimal beam width. In Figure 8, we show models' testing performance when various beam widths are used. In present KPG task with One2One (upper left), beam width of 16 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 (from beam size of 10 to 50) is not significant.

On absent KPG 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 different model architectures.

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 8 (top left), we observe that with the One2One training paradigm, a beam width of 200 does not show a significant advantage over 16, however, in terms of computation, beam width of 200 takes about 10$\times$ of the resources compared to 16. 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.

D Does Order Matter in One2Seq? — Additional Results

In §4, we show models' performance trained with the One2Seq paradigm using different target ordering strategies. Here we provide the complete set of heat maps.

In Figure 9, we show present KPG testing scores in $\mathbf{F_1@\mathcal{O}}$, when using either greedy decoding or beam search as decoding strategy.

In Figure 10, we show present KPG testing scores in $\mathbf{F_1@10}$, when using either greedy decoding or beam search as decoding strategy.

In Figure 11, we show absent KPG testing scores in **R@50**, when using either greedy decoding or beam search as decoding strategy.

In addition, we shown in Figure 12, 13, and 14 the number of unique predictions on all/present/absent KPG tasks.

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. For all experiments, we use $\mathbf{F_1@5}$, $\mathbf{F_1@10}$, $\mathbf{F_1@\mathcal{O}}$ and $\mathbf{F_1@\mathcal{M}}$ to evaluate a model's performance. Additionally, we provide an average score for each of the 4 metrics over all datasets (over each row in Table 4).

In Table 5, we report all the testing scores on absent keyphrase generation tasks. For all experiments, we use **R@10** and **R@50** to evaluate a model's performance. Additionally, we provide an average score for each of the 2 metrics over all datasets (over each row in Table 5).

In table 6, we report detailed present testing scores when model trained with One2Seq paradigm, using different ordering strategies. For all experiments, we use $\mathbf{F_1@5}$, $\mathbf{F_1@10}$, $\mathbf{F_1@\mathcal{O}}$ and $\mathbf{F_1@\mathcal{M}}$ to evaluate a model's performance.

In table 7, we report detailed absent testing scores when model trained with One2Seq paradigm, using different ordering strategies. For all experiments, we use **R@10** and **R@50** to evaluate a model's performance.

F Implementation Details

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

We will make the MAGKP dataset publicly available in the near future.

We use the concatenation of title and abstract as the source text. When training with data points contains more than 8 ground-truth keyphrases, we randomly sample 8 from the list to build the ground-truth target labels. This is to prevent training from out-of-memory issues and speed up the training.

We train RNN models for 100k steps and TRANS for 300k steps. TRANS generally benefits from longer training, especially for absent KPG performance. For the FT setting, we train modewls for additional 100k steps.

	Average				Kp20K				Krapivin				Inspec				NUS				SemEval				DUC			
Model	**5**	**10**	$\mathcal{O}$	$\mathcal{M}$	**5**	**10**	$\mathcal{O}$	$\mathcal{M}$	**5**	**10**	$\mathcal{O}$	$\mathcal{M}$	**5**	**10**	$\mathcal{O}$	$\mathcal{M}$	**5**	**10**	$\mathcal{O}$	$\mathcal{M}$	**5**	**10**	$\mathcal{O}$	$\mathcal{M}$	**5**	**10**	$\mathcal{O}$	$\mathcal{M}$
`One2One variants`																												
RNN-O2O-KP20k	30.0	28.7	32.7	15.6	33.1	27.9	35.5	11.4	32.1	26.9	36.0	11.5	29.0	32.4	33.6	23.5	40.2	36.2	43.2	18.1	34.2	35.2	34.8	19.6	11.3	13.9	13.2	9.7
RNN-O2O-KP20k-nocopy	6.2	6.5	6.1	6.6	8.3	8.5	8.4	8.5	5.6	5.8	5.0	5.8	4.9	5.3	5.3	5.4	10.3	10.9	10.4	11.1	8.4	8.4	7.5	9.0	0.0	0.0	0.0	0.0
RNN-O2O-KP20k+MagKP-ALT	31.2	30.6	34.3	13.3	32.4	27.4	34.7	9.0	32.4	28.2	35.6	9.4	32.3	37.8	38.4	21.0	40.2	38.2	43.4	15.1	35.4	35.4	37.4	16.6	14.2	16.4	16.4	8.8
BIGRNN-O2O-KP20k	31.9	31.1	35.3	13.4	35.5	29.5	38.1	9.2	34.3	29.1	38.8	9.4	31.1	36.4	37.2	20.6	42.6	39.3	45.8	15.5	34.5	36.2	37.0	17.4	13.4	16.2	15.0	8.2
BIGRNN-O2O-magkp20k-ALT	31.5	30.7	34.5	12.0	33.2	27.8	35.3	8.1	32.4	28.2	36.5	8.3	32.4	37.1	37.9	19.1	41.3	38.1	44.8	13.6	35.7	35.8	36.0	15.4	14.3	17.3	16.2	7.8
TF-O2O-KP20k	28.3	27.3	31.3	15.2	34.5	28.9	37.4	11.8	29.7	26.5	33.2	11.3	28.1	30.8	32.6	23.7	37.7	36.1	41.3	17.7	33.0	33.0	35.3	19.4	6.9	8.3	7.8	7.1
TF-O2O-KP20k-nocopy	22.0	20.5	23.9	17.4	28.5	24.0	31.2	19.0	24.7	20.9	29.0	16.0	16.6	18.2	18.4	19.0	33.7	30.2	35.3	24.0	25.3	25.9	25.8	22.1	3.4	4.1	3.7	4.2
TF-O2O-KP20k+MagKP-ALT	29.9	29.0	32.5	14.0	33.8	28.2	36.5	10.1	32.8	27.6	35.0	9.7	29.0	33.5	34.0	22.5	39.6	37.9	42.0	15.9	33.7	34.8	35.7	18.3	10.8	12.2	11.9	7.7
`One2Seq variants` (trained in Pres-Abs)																												
RNN-O2S-KP20k	29.7	29.9	32.2	22.4	31.2	26.1	31.2	16.4	31.0	27.0	33.5	18.2	32.9	38.8	38.6	32.8	37.4	36.6	39.2	25.6	33.7	35.1	36.2	26.7	11.8	15.9	14.7	14.5
RNN-O2S-KP20k-nocopy	7.1	7.0	7.5	7.0	10.4	10.2	11.1	10.2	8.1	7.9	9.9	7.9	4.5	4.5	4.5	4.5	11.0	10.5	10.9	10.6	8.8	8.6	8.8	8.6	0.1	0.1	0.1	0.1
RNN-O2S+KP20k+MagKP-ALT	28.1	29.1	31.0	22.0	28.3	23.9	28.2	15.3	28.1	25.8	30.7	17.1	32.9	40.3	39.6	33.4	35.2	33.4	36.4	23.8	30.9	33.2	34.2	26.2	13.3	18.3	17.2	16.1
BIGRNN-O2S-KP20k	28.9	29.1	31.7	22.7	30.2	25.7	30.4	16.7	29.9	26.4	32.4	18.2	31.9	37.7	37.9	32.5	37.6	35.7	39.8	27.1	32.6	33.7	35.5	27.3	11.5	15.0	14.1	14.5
BIGRNN-O2S-KP20k+MagKP-ALT	29.1	29.6	31.6	21.3	28.2	23.7	28.2	15.0	29.0	25.7	31.0	16.9	**34.8**	**41.2**	**40.1**	32.2	36.0	34.5	37.8	24.3	32.1	33.8	34.8	24.0	14.3	18.6	17.4	15.3
TF-O2S-KP20k	30.6	29.5	33.0	25.6	34.6	29.0	36.2	21.6	32.6	28.2	36.4	21.9	31.7	36.8	37.1	34.7	40.3	37.4	42.2	32.2	33.9	34.1	34.8	30.9	10.2	11.5	11.1	12.1
TF-O2S-KP20k-nocopy	25.7	24.5	27.2	23.8	32.4	29.0	33.9	**28.0**	28.5	25.1	31.6	**24.2**	23.4	24.7	25.2	24.8	37.1	34.6	37.5	**33.6**	27.4	28.4	29.5	26.9	5.3	5.1	5.2	5.2
TF-O2S-KP20k+MagKP-ALT	32.8	32.2	35.9	25.3	36.8	30.2	37.7	19.3	35.3	30.0	37.6	20.6	32.4	38.9	39.4	36.1	41.8	39.3	44.2	29.7	35.7	36.6	39.0	30.1	14.9	18.5	17.6	16.0
`TRANS+One2Seq` (trained in Pres-Abs)																												
MagKP-LN-Only	26.2	26.8	27.7	25.4	28.1	25.2	28.0	21.1	27.9	26.4	28.7	24.1	29.7	34.5	34.3	35.8	33.7	34.2	35.0	32.0	29.0	30.3	30.4	29.4	8.6	10.0	9.4	10.2
MagKP-Nsmall-Only	22.4	23.3	23.4	22.7	20.8	20.0	20.9	18.9	25.3	24.3	26.0	23.3	31.0	34.5	34.1	33.7	26.3	27.2	27.1	26.3	24.1	26.3	24.9	26.4	6.9	7.6	7.5	7.8
MagKP-Nlarge-Only	22.0	23.3	23.6	22.6	20.5	19.7	21.1	18.5	24.8	23.7	25.6	22.5	32.9	36.7	36.4	35.8	26.1	26.8	28.2	26.4	21.4	25.0	23.2	24.4	6.1	7.8	7.3	7.9
MagKP-Only	25.0	26.9	27.3	25.0	25.3	22.4	25.5	18.9	26.1	25.2	27.8	22.1	31.5	39.0	37.4	37.6	29.9	31.1	31.4	28.8	26.5	30.3	29.2	28.7	11.0	13.3	12.6	13.6
MagKP-LN-MX	31.5	30.8	34.4	24.8	35.6	29.3	36.9	19.9	34.3	28.6	37.9	20.5	31.6	38.0	37.8	35.0	41.7	38.7	44.4	30.0	32.7	35.1	34.4	29.0	13.5	15.4	14.8	14.6
MagKP-Nsmall-MX	31.1	30.8	33.6	26.5	35.4	29.2	36.5	21.4	34.2	28.8	37.0	23.0	31.9	38.4	37.3	37.5	40.6	38.7	42.6	32.8	33.2	36.1	35.4	30.7	11.1	13.4	12.5	13.3
MagKP-Nlarge-MX	31.8	31.1	34.7	26.5	36.3	30.0	37.1	21.1	35.0	29.7	37.0	23.3	31.9	37.9	38.3	38.2	42.0	39.7	44.9	31.8	34.3	35.2	37.6	30.7	11.4	13.9	13.4	13.8
MagKP-MX	32.2	32.1	34.9	**26.9**	36.3	29.8	37.4	20.2	35.1	30.1	37.2	23.0	32.1	39.8	38.5	**38.9**	41.4	40.0	44.8	31.5	34.9	36.7	36.8	31.4	13.5	16.0	14.8	16.1
MagKP-LN-FT	32.1	31.3	34.7	25.9	36.2	29.8	37.7	21.0	35.1	29.6	36.2	21.5	32.3	38.1	38.3	36.8	41.5	39.5	44.9	31.0	35.5	36.2	36.8	31.0	11.9	14.8	14.1	14.4
MagKP-Nsmall-FT	32.7	31.8	35.7	26.5	36.4	30.1	37.7	20.6	35.8	30.6	**39.4**	22.5	32.6	38.4	38.5	37.6	43.2	39.7	45.5	31.3	35.1	35.7	37.5	31.1	13.3	16.5	15.4	16.1
MagKP-Nlarge-FT	**33.6**	**32.9**	**36.6**	26.0	37.0	30.3	37.9	19.9	**36.7**	**30.8**	39.0	22.0	33.5	39.6	39.8	36.9	44.0	**40.2**	**47.8**	30.2	34.4	36.5	35.9	31.0	**16.0**	**19.7**	**19.2**	**16.2**
MagKP-FT	**33.6**	32.3	36.4	26.7	**37.1**	**30.5**	**38.3**	20.6	36.3	30.7	38.5	22.8	32.6	38.1	38.5	36.5	**44.1**	**40.2**	46.1	32.6	**36.7**	**37.1**	**39.3**	**31.9**	15.1	17.4	17.4	15.8
Abstractive Neural Generation																												
CopyRNN (Meng et al.)	-	-	-	-	32.8	25.5	-	-	30.2	25.2	-	-	29.2	33.6	-	-	34.2	31.7	-	-	29.1	29.6	-	-	-	-	-	-
CopyRNN* (Yuan et al.)	-	-	-	-	31.7	27.3	33.5	-	30.5	26.6	32.5	-	24.4	28.9	29.0	-	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	-	-	32.9	28.2	-	-	29.6	35.7	-	-	36.0	35.0	-	-	31.1	31.2	-	-	-	-	-	-
catSeqTG-2RF1† (Chan et al.)	-	-	-	-	32.1	-	-	-	30.0	-	-	-	25.3	-	-	-	37.5	-	-	-	28.7	-	-	-	-	-	-	-
KG-KE-KR-M† (Chen et al.)	-	-	-	-	31.7	28.2	-	-	27.2	25.0	-	-	25.7	28.4	-	-	28.9	28.6	-	-	20.2	22.3	-	-	-	-	-	-
CatSeq (Yuan et al.)	-	-	-	-	31.4	27.3	31.9	-	30.7	27.4	32.4	-	29.0	30.0	30.7	-	35.9	34.9	38.3	-	30.2	30.6	31.0	-	-	-	-	-
CatSeqD (Yuan et al.)	-	-	-	-	34.8	29.8	35.7	-	32.5	28.5	37.1	-	27.6	33.3	33.1	-	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.

	Average			Kp20K			Inspec			Krapivin			NUS			SemEval			DUC		
Model	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$
`One2One variants`																					
RNN-O2O-KP20k	4.4	9.0	14.0	6.8	13.2	19.7	7.9	13.9	20.9	4.5	9.3	15.0	4.7	11.3	17.6	2.2	6.1	10.0	0.0	0.0	0.6
RNN-O2O-KP20k-nocopy	1.2	3.5	6.2	2.4	5.8	9.8	1.2	4.0	8.5	1.4	2.3	3.3	1.5	5.6	9.6	0.4	3.5	6.1	0.0	0.0	0.0
RNN-O2O-KP20k+MagKP-ALT	5.2	9.9	15.2	6.6	13.3	19.5	8.4	16.3	23.9	5.7	10.6	18.1	7.0	13.8	18.8	3.2	5.2	10.0	0.0	0.1	**0.8**
BIGRNN-O2O-KP20k	7.6	14.2	19.7	11.5	20.4	28.0	13.0	22.7	31.7	6.0	12.6	18.9	10.1	18.6	24.9	4.3	10.1	13.7	**0.4**	**0.8**	**0.8**
BIGRNN-O2O-magkp20k-ALT	6.7	12.3	18.2	9.1	16.7	23.5	12.5	20.9	29.1	7.1	12.4	19.7	7.6	15.1	23.5	3.6	8.2	12.8	0.3	0.3	0.4
TF-O2O-KP20k	7.6	14.3	20.9	12.7	22.3	30.9	13.0	24.0	33.3	4.7	10.4	17.6	10.1	18.9	27.3	4.8	9.9	15.7	0.0	0.2	0.2
TF-O2O-KP20k-nocopy	8.9	**15.5**	**22.5**	14.0	**24.4**	**33.5**	15.2	**25.9**	**39.0**	5.1	10.4	14.1	**13.2**	**22.1**	**30.9**	6.1	**10.5**	**17.2**	0.0	0.0	0.1
TF-O2O-KP20k+MagKP-ALT	8.3	14.9	21.7	12.5	21.7	30.4	13.2	23.9	34.7	6.2	13.3	**21.1**	11.4	20.2	27.4	6.1	10.1	16.1	0.1	0.2	0.2
`One2Seq variants` (trained in PRES-ABS)																					
RNN-O2S-KP20k	2.2	2.5	2.5	2.7	3.2	3.3	2.8	3.2	3.3	3.5	3.8	3.8	2.6	2.9	3.0	1.6	1.7	1.7	0.0	0.0	0.0
RNN-O2S-KP20k-nocopy	3.1	4.1	4.2	4.9	6.7	6.7	4.6	6.3	6.4	1.7	2.0	2.0	4.5	6.2	6.2	2.6	3.6	3.6	0.0	0.0	0.0
RNN-O2S+KP20k+MagKP-ALT	2.0	2.7	2.8	2.6	3.2	3.3	2.4	3.2	3.4	3.7	5.1	5.1	1.4	2.5	2.6	2.0	2.0	2.1	0.0	0.0	0.0
BIGRNN-O2S-KP20k	1.7	1.8	1.8	1.9	2.0	2.0	2.6	2.6	2.6	2.5	2.6	2.6	2.2	2.3	2.3	1.0	1.0	1.0	0.3	0.3	0.3
BIGRNN-O2S-KP20k+MagKP-ALT	2.7	3.1	3.1	3.2	3.7	3.8	4.6	5.4	5.4	3.7	4.2	4.2	2.9	3.6	3.6	1.8	1.9	1.9	0.0	0.0	0.0
TF-O2S-KP20k	7.2	10.1	10.5	11.1	15.1	15.6	12.2	16.8	17.7	5.0	8.1	8.6	9.0	12.5	12.7	5.9	8.3	8.4	0.0	0.0	0.0
TF-O2S-KP20k-nocopy	8.1	11.2	11.7	13.2	18.1	18.5	14.5	20.0	21.1	5.3	7.4	7.5	10.5	15.1	15.8	5.2	6.9	7.2	0.0	0.0	0.0
TF-O2S-KP20k+MagKP-ALT	9.7	12.4	13.2	13.8	16.5	17.1	15.9	19.3	20.7	**10.8**	**15.9**	16.5	12.2	15.3	16.4	5.4	7.4	8.1	0.1	0.3	0.3
`TRANS+One2Seq +MAGKP` (trained in PRES-ABS)																					
MagKP-LN-ONLY	4.1	5.7	6.9	6.0	8.3	9.6	6.4	9.6	12.0	2.9	5.0	6.2	7.2	8.7	10.4	1.9	2.8	3.3	0.0	0.0	0.0
MagKP-Nsmall-ONLY	3.2	4.2	4.9	3.6	4.5	5.2	7.3	8.8	10.4	4.7	6.2	7.5	1.8	3.0	3.8	1.8	2.5	2.6	0.0	0.0	0.0
MagKP-Nlarge-ONLY	3.3	4.3	4.8	3.4	4.6	5.1	8.3	10.4	11.1	4.4	5.7	6.5	2.2	2.9	3.2	1.6	2.1	2.7	0.0	0.0	0.0
MagKP-ONLY	4.5	6.1	7.5	4.5	6.0	7.4	9.6	12.9	15.2	6.8	9.4	11.9	3.5	4.9	6.4	2.5	3.3	3.8	0.1	0.2	0.2
MagKP-LN-MX	8.9	11.7	12.4	13.0	16.8	17.4	15.3	19.4	19.9	6.7	10.6	11.7	12.0	15.6	17.2	6.3	8.1	8.4	0.0	0.0	0.1
MagKP-Nsmall-MX	8.9	11.7	12.8	13.1	16.8	18.0	15.2	18.7	20.7	8.1	12.7	14.5	12.2	15.2	16.2	4.3	6.1	6.9	0.1	0.4	0.4
MagKP-Nlarge-MX	9.1	11.1	12.1	13.3	15.7	16.6	15.7	19.2	21.5	8.9	12.4	13.8	11.5	13.8	14.5	5.2	5.7	6.1	0.0	0.0	0.0
MagKP-MX	9.1	10.9	12.0	12.4	14.4	15.4	15.9	17.5	19.9	10.6	14.3	16.1	10.5	12.5	13.8	5.2	6.2	6.7	0.2	0.3	0.3
MagKP-LN-FT	8.3	11.0	11.5	13.0	16.4	16.9	13.7	16.9	17.9	7.5	11.1	11.5	10.2	14.4	14.6	5.4	7.3	7.8	0.0	0.0	0.0
MagKP-Nsmall-FT	10.0	12.8	14.3	14.3	18.1	19.4	16.9	20.1	23.0	10.6	15.3	16.6	11.9	15.5	17.4	6.0	7.9	9.2	0.2	0.2	0.2
MagKP-Nlarge-FT	**10.2**	13.6	14.7	**15.1**	19.1	20.5	**17.7**	21.4	23.8	**10.8**	15.4	16.5	11.2	16.8	17.5	**6.4**	8.8	9.5	0.3	0.3	0.4
MagKP-FT	9.9	13.3	14.1	14.8	19.0	19.9	17.4	21.6	23.3	8.9	14.0	15.3	11.5	16.3	17.0	6.3	8.5	8.8	0.2	0.2	0.2

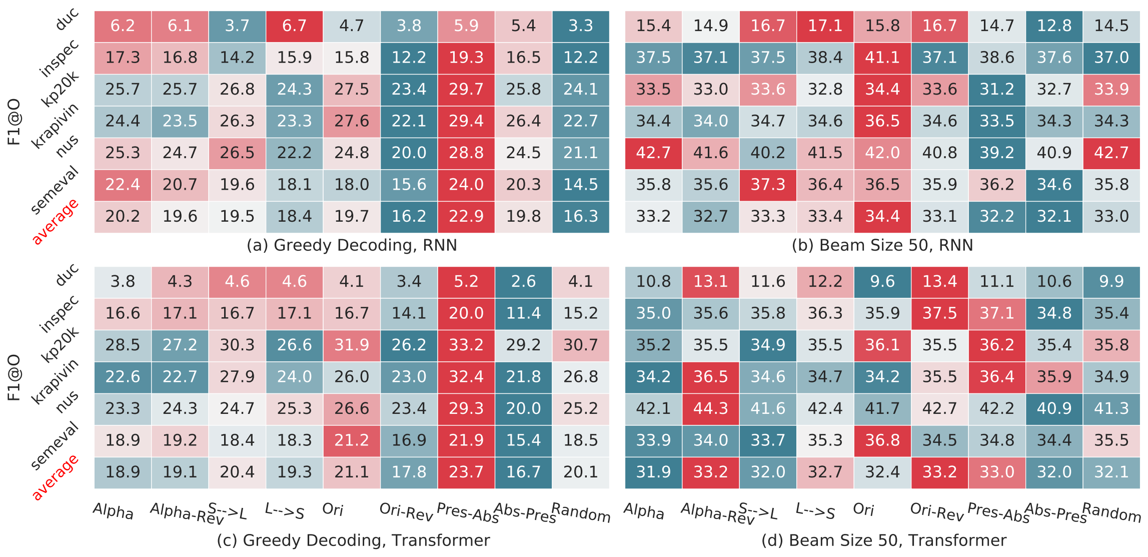
Table 5: Detailed performance (recall scores) of absent keyphrase prediction on six datasets. **Boldface**/Underline text indicates the best/2nd-best performance in corresponding columns.

	Average				Kp20K				Krapivin				Inspec				NUS				SemEval				DUC			
Model	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$
`RNN Greedy`																												
RANDOM	21.6	21.6	21.5	21.6	28.6	28.6	28.5	28.6	27.2	27.2	27.0	27.2	17.7	17.7	17.8	17.7	29.2	29.2	28.7	29.2	21.1	21.1	21.3	21.1	5.4	5.4	5.4	5.4
ALPHA	24.7	24.7	23.9	24.7	29.0	29.0	27.3	29.0	28.4	28.4	26.2	28.4	22.6	22.6	22.4	22.6	31.9	31.8	31.0	31.8	28.2	28.2	28.3	28.2	8.1	8.2	8.2	8.2
ALPHA-REV	24.5	24.5	23.4	24.5	29.4	29.4	27.5	29.4	27.9	28.0	25.5	28.0	22.4	22.3	21.9	22.3	31.1	31.1	30.6	31.1	28.0	27.9	26.4	27.9	8.2	8.2	8.2	8.2
S->L	24.1	24.1	23.9	24.1	31.0	31.0	29.8	31.0	30.2	30.2	29.3	30.2	19.6	19.6	19.4	19.6	32.0	32.0	33.1	32.0	25.8	25.8	26.0	25.8	5.6	5.6	5.6	5.6
L->S	23.6	23.6	23.1	23.6	28.1	28.1	27.0	28.1	27.7	27.7	26.3	27.7	21.7	21.7	21.7	21.7	28.9	28.9	28.7	28.9	25.2	25.2	24.9	25.2	9.8	9.8	9.8	9.8
ORI	24.4	24.4	24.5	24.4	30.5	30.5	30.7	30.5	30.6	30.6	31.0	30.6	22.0	22.0	21.9	22.0	31.8	31.8	31.7	31.8	24.5	24.5	24.6	24.5	7.3	7.3	7.3	7.3
ORI-REV	21.5	21.5	21.3	21.5	28.4	28.4	27.7	28.4	26.9	26.9	26.0	26.9	18.1	18.1	18.1	18.1	27.2	27.2	27.3	27.2	22.5	22.5	22.5	22.5	6.2	6.2	6.2	6.2
PRES-ABS	26.9	26.9	26.7	26.9	31.8	31.8	31.4	31.8	31.3	31.4	31.4	31.4	24.7	24.8	24.8	24.8	35.1	35.2	34.6	35.2	30.3	30.4	30.0	30.4	8.1	8.1	8.1	8.1
ABS-PRES	24.3	24.3	24.2	24.3	29.0	29.0	28.2	29.0	29.1	29.1	29.3	29.1	22.4	22.3	22.3	22.3	31.1	31.1	30.9	31.1	26.7	26.7	26.8	26.7	7.7	7.7	7.7	7.7
`TRANS Greedy`																												
RANDOM	24.2	24.2	24.3	24.2	33.3	33.3	33.1	33.3	28.9	28.9	29.6	28.9	20.6	20.5	20.6	20.5	31.6	31.6	31.8	31.6	24.9	24.9	25.0	24.9	5.9	5.9	5.9	5.9
ALPHA	23.5	23.5	22.4	23.5	32.8	32.9	29.8	32.9	25.9	25.8	24.2	25.8	21.8	21.8	21.8	21.8	30.2	30.2	28.9	30.2	24.9	25.0	24.9	25.0	5.2	5.2	5.2	5.2
ALPHA-REV	23.4	23.4	22.8	23.4	30.9	30.9	29.1	30.9	26.1	26.1	24.4	26.1	22.5	22.5	22.5	22.5	29.8	29.7	29.8	29.7	25.0	25.0	24.9	25.0	6.0	6.0	6.0	6.0
S->L	24.7	24.7	24.3	24.7	33.3	33.3	31.9	33.3	31.0	31.0	30.0	31.0	21.9	21.9	22.0	21.9	30.4	30.4	30.8	30.4	25.2	25.2	24.8	25.2	6.3	6.3	6.3	6.3
L->S	23.5	23.5	23.3	23.5	29.8	29.8	28.9	29.8	26.9	26.9	26.2	26.9	23.2	23.2	23.2	23.2	31.0	30.9	31.5	30.9	23.6	23.6	23.8	23.6	6.3	6.3	6.3	6.3
ORI	24.5	24.5	24.7	24.5	32.8	32.8	33.4	32.8	27.6	27.6	27.7	27.6	22.3	22.3	22.3	22.3	31.6	31.6	31.9	31.6	26.7	26.7	27.1	26.7	5.9	5.9	5.9	5.9
ORI-REV	22.7	22.7	22.5	22.7	30.6	30.6	29.5	30.6	26.4	26.4	26.0	26.4	19.7	19.7	19.8	19.7	30.2	30.2	30.5	30.2	24.1	24.1	23.8	24.1	5.3	5.3	5.3	5.3
PRES-ABS	26.8	26.8	27.5	26.8	34.0	34.0	34.7	34.0	31.8	31.9	33.8	31.9	25.8	25.8	25.7	25.8	34.2	34.3	35.2	34.3	28.0	28.0	28.4	28.0	6.9	7.0	7.0	7.0
ABS-PRES	20.3	20.3	20.3	20.3	31.2	31.3	31.0	31.3	23.8	23.8	24.3	23.8	16.1	16.1	16.1	16.1	25.8	25.8	25.6	25.8	21.2	21.2	21.3	21.2	3.7	3.7	3.7	3.7
`RNN beam50`																												
RANDOM	30.3	29.3	33.0	24.4	31.9	25.9	33.9	19.6	32.0	26.8	34.3	20.5	31.9	36.3	37.0	33.2	39.2	35.4	42.7	28.9	34.5	35.9	35.8	29.8	12.4	15.6	14.5	14.3
ALPHA	30.8	30.2	33.2	22.6	32.6	27.1	33.5	17.4	33.0	28.1	34.4	18.9	31.8	37.5	37.5	31.3	39.6	37.4	42.7	26.7	34.5	35.2	35.8	26.9	13.5	15.9	15.4	14.1
ALPHA-REV	30.3	29.7	32.7	23.3	32.6	27.2	33.0	18.5	32.1	27.6	34.0	19.5	30.9	36.1	37.1	31.8	39.8	37.4	41.6	27.9	33.3	33.9	35.6	28.4	13.2	16.0	14.9	13.6
S->L	30.8	29.6	33.3	21.2	32.6	26.6	33.6	17.0	33.4	27.3	34.7	18.3	32.5	36.6	37.5	29.5	38.4	35.3	40.2	24.3	33.9	34.7	37.3	25.4	14.2	17.0	16.7	13.0
L->S	31.4	30.6	33.4	27.7	32.4	27.4	32.8	22.6	33.5	28.2	34.6	23.9	33.3	38.1	38.4	36.1	38.5	37.5	41.5	33.0	35.4	34.4	36.4	33.3	15.1	17.9	17.1	17.2
ORI	31.6	30.4	34.4	23.2	32.4	26.2	34.4	17.2	33.1	28.0	36.5	19.2	34.5	39.8	41.1	34.3	40.0	36.3	42.0	26.8	35.5	35.3	36.5	27.4	14.0	16.8	15.8	14.6
ORI-REV	31.1	29.6	33.1	26.6	32.0	26.9	33.6	22.9	32.3	27.5	34.6	23.5	33.0	35.5	37.1	32.4	39.6	36.4	40.8	32.8	35.2	34.1	35.9	31.8	14.7	17.0	16.7	16.4
PRES-ABS	29.7	29.9	32.2	22.4	31.2	26.1	31.2	16.4	31.0	27.0	33.5	18.2	32.9	38.8	38.6	32.8	37.4	36.6	39.2	25.6	33.7	35.1	36.2	26.7	11.8	15.9	14.7	14.5
ABS-PRES	29.1	29.9	32.1	19.7	31.7	27.0	32.7	13.8	31.0	27.7	34.3	15.5	31.7	37.1	37.6	29.9	38.3	37.5	40.9	22.8	31.5	35.3	34.6	23.5	10.5	14.5	12.8	12.7
`TRANS beam50`																												
RANDOM	30.3	28.8	32.1	25.4	34.4	28.2	35.8	22.8	33.9	29.0	34.9	23.8	31.3	34.9	35.4	32.9	38.9	36.9	41.3	32.4	34.0	33.4	35.5	29.8	9.3	10.5	9.9	10.7
ALPHA	30.4	28.9	31.9	25.2	34.7	29.2	35.2	22.8	33.4	27.6	34.2	21.5	30.9	34.3	35.0	33.2	39.9	37.4	42.1	32.4	33.8	33.9	33.9	30.6	9.9	11.2	10.8	11.1
ALPHA-REV	30.3	29.8	33.2	25.4	34.7	29.0	35.5	21.8	32.7	27.5	36.5	21.7	30.4	36.1	35.6	34.0	40.4	37.6	44.3	31.4	31.6	34.8	34.0	30.6	12.0	13.8	13.1	12.7
S->L	30.4	29.3	32.0	25.6	34.4	28.5	34.9	22.2	33.7	28.5	34.6	23.5	31.1	35.6	35.8	33.1	40.5	37.9	41.6	32.3	32.6	34.0	33.7	31.2	10.1	11.6	11.6	11.4
L->S	30.8	29.9	32.7	25.9	34.7	29.0	35.5	22.6	33.5	29.0	34.7	22.7	31.8	35.6	36.3	33.6	40.7	38.4	42.4	32.5	33.5	34.3	35.3	31.5	10.6	12.9	12.2	12.7
ORI	30.2	29.4	32.4	25.3	34.4	28.5	36.1	21.8	33.6	29.2	34.2	23.2	31.9	36.6	35.9	34.1	38.5	37.6	41.7	31.9	34.3	34.1	36.8	29.9	8.7	10.2	9.6	10.6
ORI-REV	31.0	30.1	33.2	27.1	34.6	28.9	35.5	24.1	33.9	28.7	35.5	24.5	32.7	36.3	37.5	33.8	40.1	38.3	42.7	34.3	32.5	34.8	34.5	32.3	11.9	13.8	13.4	13.5
PRES-ABS	30.6	29.5	33.0	25.6	34.6	29.0	36.2	21.6	32.6	28.2	36.4	21.9	31.7	36.8	37.1	34.7	40.3	37.4	42.2	32.2	33.9	34.1	34.8	30.9	10.2	11.5	11.1	12.1
ABS-PRES	29.7	29.3	32.0	22.3	34.3	28.9	35.4	17.9	32.5	28.0	35.9	18.0	30.7	34.9	34.8	32.5	38.4	38.4	40.9	26.9	33.1	34.5	34.4	27.9	9.4	11.1	10.6	10.3

Table 6: Detailed present keyphrase prediction performance (F_1-score) of `One2Seq` trained with different orders.

	Average			Kp20K			Inspec			Krapivin			NUS			SemEval			DUC		
Model	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$
`RNN Greedy`																					
RANDOM	0.2	0.2	0.2	0.2	0.2	0.2	0.4	0.4	0.4	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3	0.0	0.0	0.0
ALPHA	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.0	0.0	0.0	0.7	0.7	0.7	0.0	0.0	0.0
ALPHA-REV	0.2	0.2	0.2	0.2	0.2	0.2	0.4	0.4	0.4	0.1	0.1	0.1	0.1	0.1	0.1	0.3	0.3	0.3	0.0	0.0	0.0
S->L	0.1	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.0	0.0	0.0
L->S	0.2	0.2	0.2	0.2	0.2	0.2	0.6	0.6	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.6	0.6	0.0	0.0	0.0
ORI	0.3	0.3	0.3	0.2	0.2	0.2	0.6	0.6	0.6	0.1	0.1	0.1	0.2	0.2	0.2	0.5	0.5	0.5	0.0	0.0	0.0
ORI-REV	0.1	0.1	0.1	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.3	0.3	0.0	0.0	0.0
PRES-ABS	0.2	0.2	0.2	0.2	0.2	0.2	0.5	0.5	0.5	0.3	0.3	0.3	0.2	0.2	0.2	0.3	0.3	0.3	0.0	0.0	0.0
ABS-PRES	0.2	0.2	0.2	0.3	0.3	0.3	0.4	0.4	0.4	0.3	0.3	0.3	0.0	0.0	0.0	0.4	0.4	0.4	0.0	0.0	0.0
`TRANS Greedy`																					
RANDOM	1.9	1.9	1.9	2.8	2.8	2.8	2.4	2.4	2.4	0.8	0.8	0.8	3.1	3.1	3.1	2.1	2.1	2.1	0.0	0.0	0.0
ALPHA	1.6	1.6	1.6	3.0	3.0	3.0	3.6	3.6	3.6	0.8	0.8	0.8	0.9	0.9	0.9	1.1	1.1	1.1	0.0	0.0	0.0
ALPHA-REV	1.6	1.6	1.6	2.5	2.5	2.5	3.9	3.9	3.9	1.1	1.1	1.1	1.4	1.4	1.4	0.7	0.7	0.7	0.0	0.0	0.0
S->L	1.5	1.5	1.5	2.8	2.8	2.8	2.5	2.5	2.5	0.4	0.4	0.4	1.8	1.8	1.8	1.3	1.3	1.3	0.0	0.0	0.0
L->S	1.3	1.3	1.3	2.4	2.4	2.4	2.4	2.4	2.4	0.6	0.6	0.6	1.6	1.6	1.6	0.8	0.8	0.8	0.0	0.0	0.0
ORI	1.5	1.5	1.5	2.5	2.5	2.5	3.3	3.3	3.3	1.1	1.1	1.1	1.2	1.2	1.2	0.9	0.9	0.9	0.1	0.1	0.1
ORI-REV	1.1	1.1	1.1	2.5	2.5	2.5	2.2	2.2	2.2	0.5	0.5	0.5	0.8	0.8	0.8	0.8	0.8	0.8	0.0	0.0	0.0
PRES-ABS	1.3	1.3	1.3	2.3	2.3	2.3	2.7	2.7	2.7	0.8	0.8	0.8	1.0	1.0	1.0	1.2	1.2	1.2	0.0	0.0	0.0
ABS-PRES	1.8	1.8	1.8	2.8	2.8	2.8	4.0	4.0	4.0	0.7	0.7	0.7	2.0	2.0	2.0	1.6	1.6	1.6	0.0	0.0	0.0
`RNN beam50`																					
RANDOM	1.9	2.1	2.1	2.4	2.7	2.7	2.4	2.7	2.7	3.3	3.6	3.6	1.5	1.9	1.9	1.5	1.6	1.6	0.0	0.0	0.0
ALPHA	2.1	2.4	2.4	2.5	2.7	2.7	2.9	3.1	3.1	3.4	3.8	3.8	2.4	2.6	2.6	1.8	2.0	2.0	0.0	0.0	0.0
ALPHA-REV	1.4	1.5	1.5	2.0	2.1	2.1	2.1	2.2	2.2	1.7	1.7	1.7	2.1	2.2	2.2	0.6	0.8	0.8	0.0	0.0	0.0
S->L	1.7	1.9	1.9	2.1	2.2	2.2	2.1	2.2	2.2	2.8	3.6	3.6	1.3	1.4	1.4	1.9	2.0	2.0	0.0	0.0	0.2
L->S	1.4	1.4	1.4	1.8	1.8	1.8	2.1	2.1	2.1	2.5	2.5	2.5	1.1	1.1	1.1	0.8	0.8	0.8	0.0	0.0	0.0
ORI	1.9	2.3	2.3	2.3	2.7	2.7	2.7	3.2	3.2	3.0	3.6	3.6	2.3	2.8	2.8	1.2	1.9	1.9	0.0	0.0	0.0
ORI-REV	1.5	1.6	1.6	1.9	2.0	2.0	2.0	2.0	2.0	2.5	2.5	2.5	1.3	1.4	1.4	1.5	1.6	1.6	0.0	0.0	0.0
PRES-ABS	2.2	2.5	2.5	2.7	3.2	3.3	2.8	3.2	3.3	3.5	3.8	3.8	2.6	2.9	3.0	1.6	1.7	1.7	0.0	0.0	0.0
ABS-PRES	1.1	1.1	1.1	1.2	1.3	1.3	1.5	1.5	1.5	1.6	1.6	1.6	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0
`TRANS beam50`																					
RANDOM	6.7	8.0	8.2	10.2	12.5	12.8	11.5	13.1	13.3	3.8	5.7	5.8	10.0	11.2	11.5	4.9	5.7	6.0	0.1	0.1	0.1
ALPHA	7.3	9.6	9.7	11.8	14.8	14.9	12.9	17.2	17.3	5.0	7.2	7.5	9.1	11.5	11.5	5.2	6.7	6.7	0.0	0.1	0.1
ALPHA-REV	6.5	9.7	9.8	10.3	14.3	14.6	10.2	15.4	15.6	3.8	5.7	5.9	8.9	15.6	15.8	5.7	7.2	7.2	0.0	0.0	0.0
S->L	6.5	8.5	8.6	10.3	13.1	13.2	11.0	14.2	14.3	3.7	6.0	6.0	9.0	11.0	11.3	5.0	6.6	6.9	0.0	0.0	0.0
L->S	6.8	9.5	9.8	11.1	15.3	15.7	11.6	15.1	15.6	4.2	6.7	7.4	9.2	12.7	12.8	4.9	7.1	7.2	0.0	0.0	0.0
ORI	7.0	10.0	10.4	11.2	15.3	15.7	12.3	16.9	17.6	4.1	6.4	6.7	9.9	14.4	14.9	4.5	7.0	7.3	0.1	0.1	0.1
ORI-REV	7.2	10.4	10.8	11.3	16.0	16.6	12.2	16.9	17.3	4.6	7.2	7.9	9.8	13.7	14.3	5.3	8.2	8.3	0.0	0.3	0.3
PRES-ABS	7.2	10.1	10.5	11.1	15.1	15.6	12.2	16.8	17.7	5.0	8.1	8.6	9.0	12.5	12.7	5.9	8.3	8.4	0.0	0.0	0.0
ABS-PRES	6.2	7.0	7.1	10.2	11.4	11.4	10.6	11.3	11.3	4.2	5.7	5.9	7.5	7.8	7.8	4.9	6.0	6.0	0.0	0.0	0.0

Table 7: Detailed absent keyphrase prediction performance (recall scores) of `One2Seq` trained with different orders.



F1@O

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	6.2	6.1	3.7	6.7	4.7	3.8	5.9	5.4	3.3
inspec	17.3	16.8	14.2	15.9	15.8	12.2	19.3	16.5	12.2
kp20k	25.7	25.7	26.8	24.3	27.5	23.4	29.7	25.8	24.1
krapivin	24.4	23.5	26.3	23.3	27.6	22.1	29.4	26.4	22.7
nus	25.3	24.7	26.5	22.2	24.8	20.0	28.8	24.5	21.1
semeval	22.4	20.7	19.6	18.1	18.0	15.6	24.0	20.3	14.5
average	20.2	19.6	19.5	18.4	19.7	16.2	22.9	19.8	16.3

(a) Greedy Decoding, RNN

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	15.4	14.9	16.7	17.1	15.8	16.7	14.7	12.8	14.5
inspec	37.5	37.1	37.5	38.4	41.1	37.1	38.6	37.6	37.0
kp20k	33.5	33.0	33.6	32.8	34.4	33.6	31.2	32.7	33.9
krapivin	34.4	34.0	34.7	34.6	36.5	34.6	33.5	34.3	34.3
nus	42.7	41.6	40.2	41.5	42.0	40.8	39.2	40.9	42.7
semeval	35.8	35.6	37.3	36.4	36.5	35.9	36.2	34.6	35.8
average	33.2	32.7	33.3	33.4	34.4	33.1	32.2	32.1	33.0

(b) Beam Size 50, RNN

F1@O

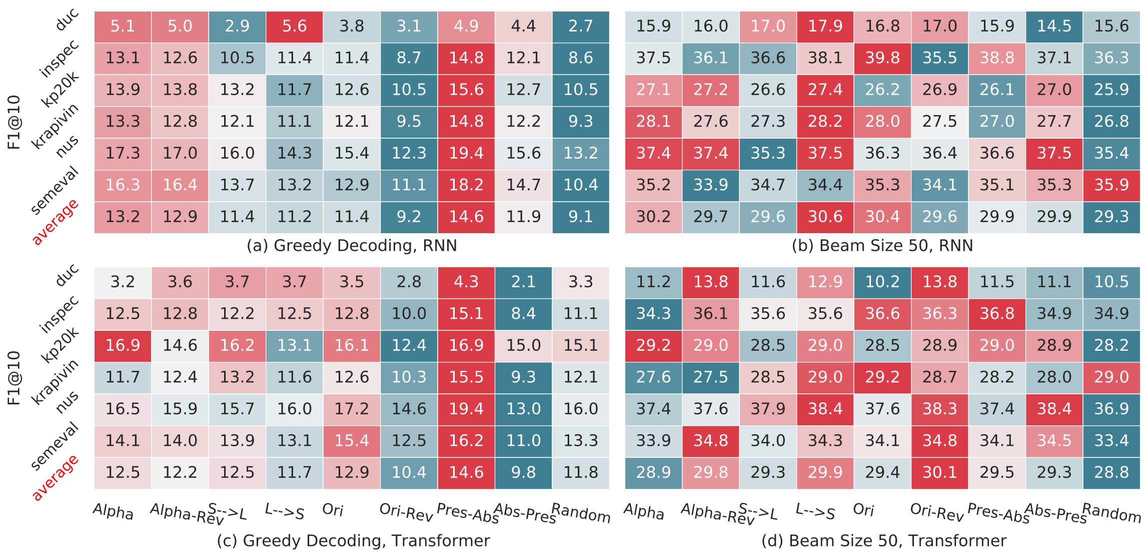
	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	3.8	4.3	4.6	4.6	4.1	3.4	5.2	2.6	4.1
inspec	16.6	17.1	16.7	17.1	16.7	14.1	20.0	11.4	15.2
kp20k	28.5	27.2	30.3	26.6	31.9	26.2	33.2	29.2	30.7
krapivin	22.6	22.7	27.9	24.0	26.0	23.0	32.4	21.8	26.8
nus	23.3	24.3	24.7	25.3	26.6	23.4	29.3	20.0	25.2
semeval	18.9	19.2	18.4	18.3	21.2	16.9	21.9	15.4	18.5
average	18.9	19.1	20.4	19.3	21.1	17.8	23.7	16.7	20.1

(c) Greedy Decoding, Transformer

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	10.8	13.1	11.6	12.2	9.6	13.4	11.1	10.6	9.9
inspec	35.0	35.6	35.8	36.3	35.9	37.5	37.1	34.8	35.4
kp20k	35.2	35.5	34.9	35.5	36.1	35.5	36.2	35.4	35.8
krapivin	34.2	36.5	34.6	34.7	34.2	35.5	36.4	35.9	34.9
nus	42.1	44.3	41.6	42.4	41.7	42.7	42.2	40.9	41.3
semeval	33.9	34.0	33.7	35.3	36.8	34.5	34.8	34.4	35.5
average	31.9	33.2	32.0	32.7	32.4	33.2	33.0	32.0	32.1

(d) Beam Size 50, Transformer

Figure 9: Present keyphrase generation testing scores ($\mathbf{F}_1@\mathcal{O}$). Colors represent the relative performance, normalized per row.



F1@10

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	5.1	5.0	2.9	5.6	3.8	3.1	4.9	4.4	2.7
inspec	13.1	12.6	10.5	11.4	11.4	8.7	14.8	12.1	8.6
kp20k	13.9	13.8	13.2	11.7	12.6	10.5	15.6	12.7	10.5
krapivin	13.3	12.8	12.1	11.1	12.1	9.5	14.8	12.2	9.3
nus	17.3	17.0	16.0	14.3	15.4	12.3	19.4	15.6	13.2
semeval	16.3	16.4	13.7	13.2	12.9	11.1	18.2	14.7	10.4
average	13.2	12.9	11.4	11.2	11.4	9.2	14.6	11.9	9.1

(a) Greedy Decoding, RNN

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	15.9	16.0	17.0	17.9	16.8	17.0	15.9	14.5	15.6
inspec	37.5	36.1	36.6	38.1	39.8	35.5	38.8	37.1	36.3
kp20k	27.1	27.2	26.6	27.4	26.2	26.9	26.1	27.0	25.9
krapivin	28.1	27.6	27.3	28.2	28.0	27.5	27.0	27.7	26.8
nus	37.4	37.4	35.3	37.5	36.3	36.4	36.6	37.5	35.4
semeval	35.2	33.9	34.7	34.4	35.3	34.1	35.1	35.3	35.9
average	30.2	29.7	29.6	30.6	30.4	29.6	29.9	29.9	29.3

(b) Beam Size 50, RNN

F1@10

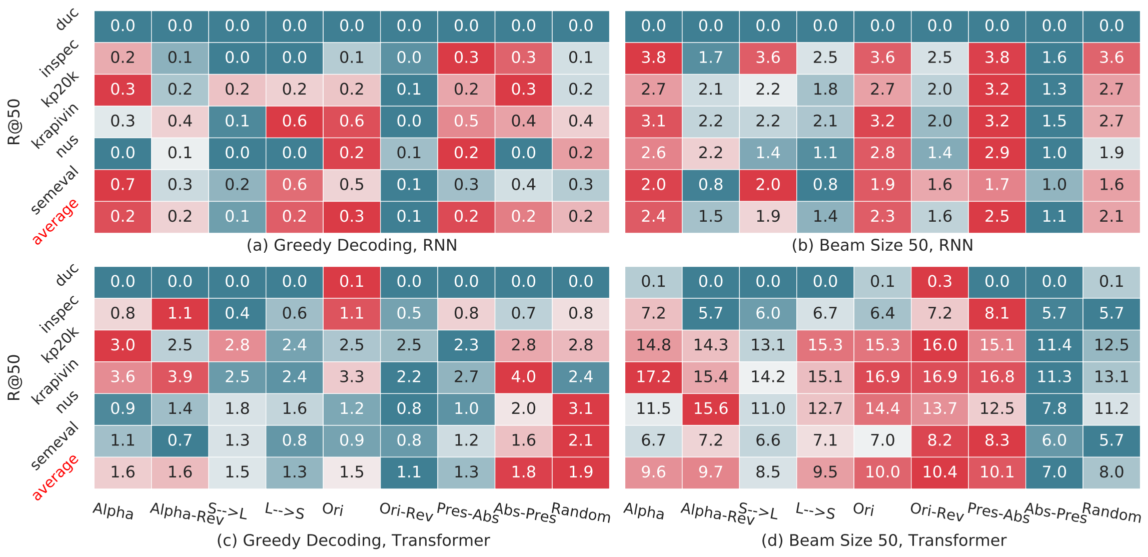
	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	3.2	3.6	3.7	3.7	3.5	2.8	4.3	2.1	3.3
inspec	12.5	12.8	12.2	12.5	12.8	10.0	15.1	8.4	11.1
kp20k	16.9	14.6	16.2	13.1	16.1	12.4	16.9	15.0	15.1
krapivin	11.7	12.4	13.2	11.6	12.6	10.3	15.5	9.3	12.1
nus	16.5	15.9	15.7	16.0	17.2	14.6	19.4	13.0	16.0
semeval	14.1	14.0	13.9	13.1	15.4	12.5	16.2	11.0	13.3
average	12.5	12.2	12.5	11.7	12.9	10.4	14.6	9.8	11.8

(c) Greedy Decoding, Transformer

	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	11.2	13.8	11.6	12.9	10.2	13.8	11.5	11.1	10.5
inspec	34.3	36.1	35.6	35.6	36.6	36.3	36.8	34.9	34.9
kp20k	29.2	29.0	28.5	29.0	28.5	28.9	29.0	28.9	28.2
krapivin	27.6	27.5	28.5	29.0	29.2	28.7	28.2	28.0	29.0
nus	37.4	37.6	37.9	38.4	37.6	38.3	37.4	38.4	36.9
semeval	33.9	34.8	34.0	34.3	34.1	34.8	34.1	34.5	33.4
average	28.9	29.8	29.3	29.9	29.4	30.1	29.5	29.3	28.8

(d) Beam Size 50, Transformer

Figure 10: Present keyphrase generation testing scores ($\mathbf{F}_1$@10). Colors represent the relative performance, normalized per row.



R@50	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
inspec	0.2	0.1	0.0	0.0	0.1	0.0	0.3	0.3	0.1
kp20k	0.3	0.2	0.2	0.2	0.2	0.1	0.2	0.3	0.2
krapivin	0.3	0.4	0.1	0.6	0.6	0.0	0.5	0.4	0.4
nus	0.0	0.1	0.0	0.0	0.2	0.1	0.2	0.0	0.2
semeval	0.7	0.3	0.2	0.6	0.5	0.1	0.3	0.4	0.3
average	0.2	0.2	0.1	0.2	0.3	0.1	0.2	0.2	0.2

(a) Greedy Decoding, RNN

R@50	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
inspec	3.8	1.7	3.6	2.5	3.6	2.5	3.8	1.6	3.6
kp20k	2.7	2.1	2.2	1.8	2.7	2.0	3.2	1.3	2.7
krapivin	3.1	2.2	2.2	2.1	3.2	2.0	3.2	1.5	2.7
nus	2.6	2.2	1.4	1.1	2.8	1.4	2.9	1.0	1.9
semeval	2.0	0.8	2.0	0.8	1.9	1.6	1.7	1.0	1.6
average	2.4	1.5	1.9	1.4	2.3	1.6	2.5	1.1	2.1

(b) Beam Size 50, RNN

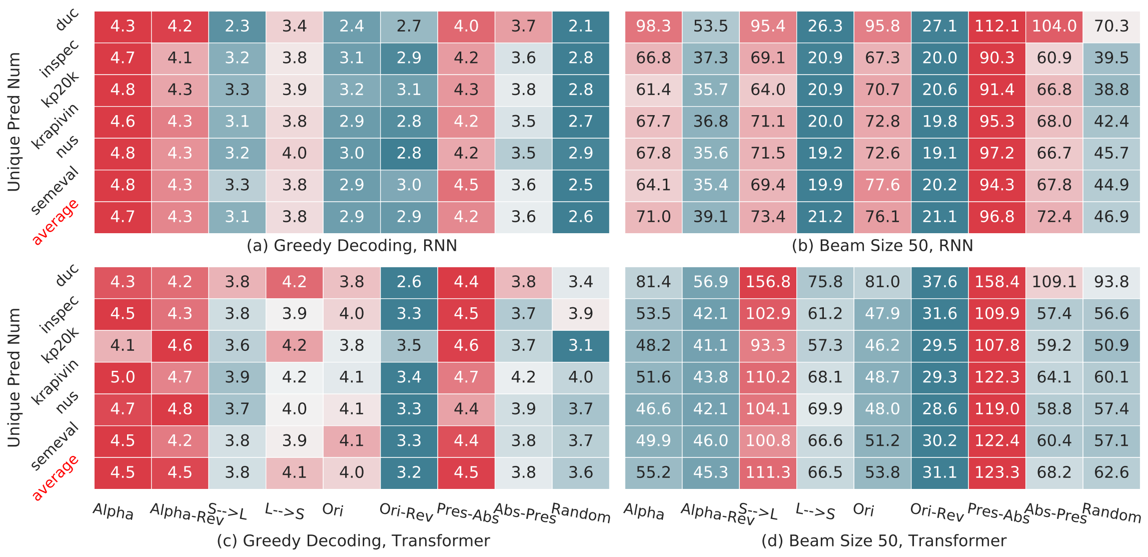
R@50	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
inspec	0.8	1.1	0.4	0.6	1.1	0.5	0.8	0.7	0.8
kp20k	3.0	2.5	2.8	2.4	2.5	2.5	2.3	2.8	2.8
krapivin	3.6	3.9	2.5	2.4	3.3	2.2	2.7	4.0	2.4
nus	0.9	1.4	1.8	1.6	1.2	0.8	1.0	2.0	3.1
semeval	1.1	0.7	1.3	0.8	0.9	0.8	1.2	1.6	2.1
average	1.6	1.6	1.5	1.3	1.5	1.1	1.3	1.8	1.9

(c) Greedy Decoding, Transformer

R@50	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	0.1	0.0	0.0	0.0	0.1	0.3	0.0	0.0	0.1
inspec	7.2	5.7	6.0	6.7	6.4	7.2	8.1	5.7	5.7
kp20k	14.8	14.3	13.1	15.3	15.3	16.0	15.1	11.4	12.5
krapivin	17.2	15.4	14.2	15.1	16.9	16.9	16.8	11.3	13.1
nus	11.5	15.6	11.0	12.7	14.4	13.7	12.5	7.8	11.2
semeval	6.7	7.2	6.6	7.1	7.0	8.2	8.3	6.0	5.7
average	9.6	9.7	8.5	9.5	10.0	10.4	10.1	7.0	8.0

(d) Beam Size 50, Transformer

Figure 11: Absent keyphrase generation testing scores on **R@50**. Colors represent the relative performance, normalized per row.



Unique Pred Num	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	4.3	4.2	2.3	3.4	2.4	2.7	4.0	3.7	2.1
inspec	4.7	4.1	3.2	3.8	3.1	2.9	4.2	3.6	2.8
kp20k	4.8	4.3	3.3	3.9	3.2	3.1	4.3	3.8	2.8
krapivin	4.6	4.3	3.1	3.8	2.9	2.8	4.2	3.5	2.7
nus	4.8	4.3	3.2	4.0	3.0	2.8	4.2	3.5	2.9
semeval	4.8	4.3	3.3	3.8	2.9	3.0	4.5	3.6	2.5
average	4.7	4.3	3.1	3.8	2.9	2.9	4.2	3.6	2.6

(a) Greedy Decoding, RNN

Unique Pred Num	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	98.3	53.5	95.4	26.3	95.8	27.1	112.1	104.0	70.3
inspec	66.8	37.3	69.1	20.9	67.3	20.0	90.3	60.9	39.5
kp20k	61.4	35.7	64.0	20.9	70.7	20.6	91.4	66.8	38.8
krapivin	67.7	36.8	71.1	20.0	72.8	19.8	95.3	68.0	42.4
nus	67.8	35.6	71.5	19.2	72.6	19.1	97.2	66.7	45.7
semeval	64.1	35.4	69.4	19.9	77.6	20.2	94.3	67.8	44.9
average	71.0	39.1	73.4	21.2	76.1	21.1	96.8	72.4	46.9

(b) Beam Size 50, RNN

Unique Pred Num	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	4.3	4.2	3.8	4.2	3.8	2.6	4.4	3.8	3.4
inspec	4.5	4.3	3.8	3.9	4.0	3.3	4.5	3.7	3.9
kp20k	4.1	4.6	3.6	4.2	3.8	3.5	4.6	3.7	3.1
krapivin	5.0	4.7	3.9	4.2	4.1	3.4	4.7	4.2	4.0
nus	4.7	4.8	3.7	4.0	4.1	3.3	4.4	3.9	3.7
semeval	4.5	4.2	3.8	3.9	4.1	3.3	4.4	3.8	3.7
average	4.5	4.5	3.8	4.1	4.0	3.2	4.5	3.8	3.6

(c) Greedy Decoding, Transformer

Unique Pred Num	Alpha	Alpha-Rev	S-->L	L-->S	Ori	Ori-Rev	Pres-Abs	Abs-Pres	Random
duc	81.4	56.9	156.8	75.8	81.0	37.6	158.4	109.1	93.8
inspec	53.5	42.1	102.9	61.2	47.9	31.6	109.9	57.4	56.6
kp20k	48.2	41.1	93.3	57.3	46.2	29.5	107.8	59.2	50.9
krapivin	51.6	43.8	110.2	68.1	48.7	29.3	122.3	64.1	60.1
nus	46.6	42.1	104.1	69.9	48.0	28.6	119.0	58.8	57.4
semeval	49.9	46.0	100.8	66.6	51.2	30.2	122.4	60.4	57.1
average	55.2	45.3	111.3	66.5	53.8	31.1	123.3	68.2	62.6

(d) Beam Size 50, Transformer

Figure 12: Unique number of keyphrases generated during test. Colors represent the relative performance, normalized per row.

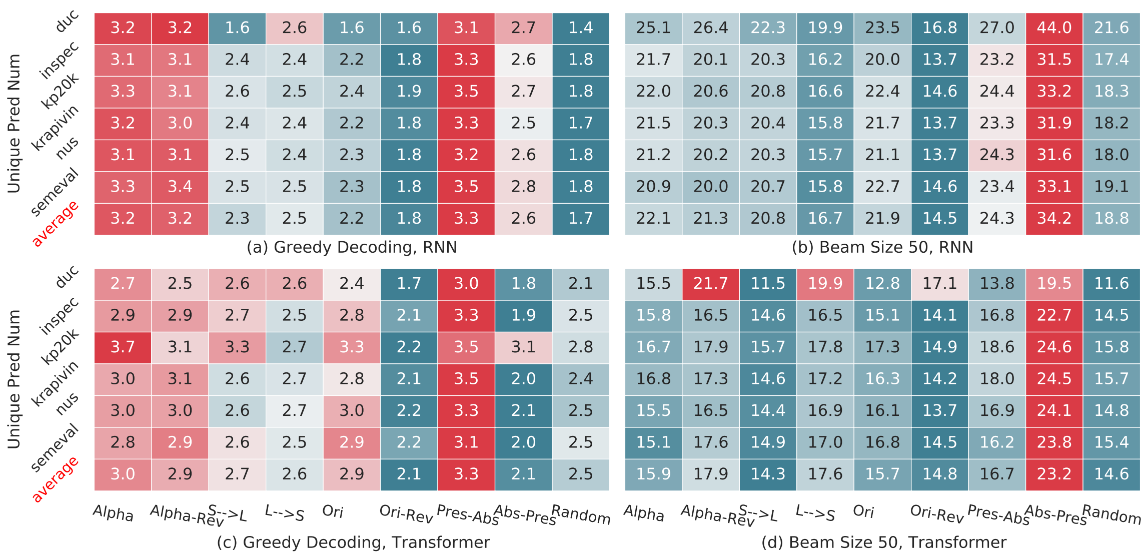


Figure 13: Unique number of present keyphrases generated during test. Colors represent the relative performance, normalized per row.

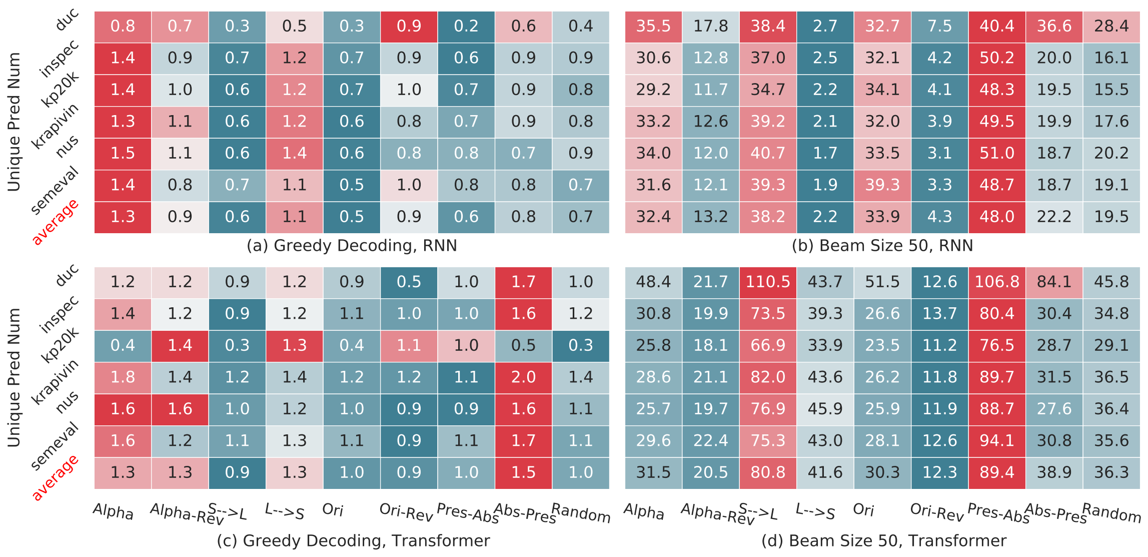


Figure 14: Unique number of absent keyphrases generated during test. Colors represent the relative performance, normalized per row.