Neural Keyphrase Generation: Analysis and Evaluation

Anonymous ACL submission

Abstract

Keyphrase generation aims at generating topical phrases from a given text either by copying from the original text (present keyphrases) or by producing new keyphrases (absent keyphrases) that capture the semantic meaning of the text. Encoder-decoder models are most widely used for this task because of their capabilities for absent keyphrase generation. However, there has been little to no analysis on the performance and behavior of such models for keyphrase generation. In this paper, we study various tendencies exhibited by two strong models: T5 (based on a pre-trained transformer) and ExHiRD (based on a recurrent neural network). We analyze prediction confidence scores, model calibration, and the effect of position on present keyphrases generation. Moreover, we motivate and propose a novel metric, SoftKeyScore, to evaluate the similarity between two sets of keyphrases by using soft-scores to account for partial matching and semantic similarity. We find that SoftKeyScore performs better than the standard F_1 metric for evaluating two sets of given keyphrases. We will release our code.

1 Introduction

Keyphrases are phrases that capture the core ideas and topics of a given document. Keyphrase generation is the task of predicting a set of keyphrases from a given document. Among these keyphrases, some exist within the source document (present keyphrases), and some are absent from the document (absent keyphrases). Keyphrases are widely used in various applications, such as document indexing and retrieval (Jones and Staveley, 1999; Boudin et al., 2020), document clustering (Hulth and Megyesi, 2006), and text summarization (Wang and Cardie, 2013). Hence, keyphrase generation is of great interest to the scientific community.

In recent years, neural encoder-decoder (seq2seq) models are adapted to generate both absent and present keyphrases (Meng et al., 2017). Most contemporary approaches (Yuan et al., 2020; Chan et al., 2019a; Chen et al., 2020) to keyphrase generation aim at autoregressively decoding a sequence of concatenated keyphrases from a given source document. Typically, these models are equipped with cross-attention (Luong et al., 2015; Bahdanau et al., 2015) and a copy (or pointer) mechanism (Gu et al., 2016; See et al., 2017). Although several variants and extensions of seq2seq models have been proposed to enhance keyphrase generation (Meng et al., 2017; Yuan et al., 2020; Chan et al., 2019a; Swaminathan et al., 2020; Chen et al., 2020), there have been limited attempts at deeper analysis on the tendencies of the neural seq2seq in this task.

Moreover, despite the ubiquitous success of pre-trained models (typically Transformers) on several NLP tasks, there is a dearth of exploration of pre-trained models for keyphrase generation. While most pre-trained models such as BERT (Devlin et al., 2019) and ELECTRA (Clark et al., 2020) are focused on encoding, recently there have been a few pre-trained seq2seq transformers (e.g., T5, BART, and PEGASUS) (Raffel et al., 2020; Lewis et al., 2020; Zhang et al., 2020) which are natural choices to be adapted for keyphrase generation.

In this work, we explore T5[1] (Raffel et al., 2020), a pre-trained seq2seq Transformer, and contrast its performance with a strong recurrent neural network (RNN) based seq2seq architecture for keyphrase generation (ExHiRD) (Chen et al., 2020) on different aspects of the task.

Overall, our contributions are as follows:

1. We introduce keyphrase perplexity (KPP) to gauge model confidence. Using KPP, we analyze the prediction confidence of a pre-trained

[1]Interestingly, the pre-training objective in T5 for generating a series of concatenated spans which are masked in the source text also happens to be particularly similar to the downstream task of our desire (keyphrase generation).

model (T5) and a trained-from-scratch RNN-based seq2seq model (ExHiRD). In addition, we explore the models' calibration to study confidence versus generation performance.

2. We empirically evaluate and contrast the performance of T5 and ExHiRD, on standard F1-based measures.
3. We examine the variance of model performance with that of the position of extracted present keyphrases in the source document.
4. We propose an evaluation framework, SoftKeyScore, to measure the similarity of two sets of keyphrases (the predicted set and the gold set) using soft-scoring functions to account for partial matches and semantic similarities between predicted keyphrases and target keyphrases. We perform evaluation to verify the correlation of the various evaluation metrics against human annotated scores.

2 Related Work

The current focus of research on keyphrase generation has been increasingly shifting towards the use of neural generative (sequence-to-sequence) models (Meng et al., 2017) particularly because of their capability to generate absent keyphrases. Meng et al. (2017) used a Recurrent Neural Network (RNN) model along with CopyNet (CopyRNN) for keyphrase generation. Chen et al. (2018) extended CopyRNN by utilizing correlations between the predicted keyphrases. Chen et al. (2019) introduced a title-guided encoding scheme in a seq2seq model. All these methods, however, could only predict one keyphrase and they had to rely on beam search to predict more keyphrases. Yuan et al. (2020) solved this issue by allowing their model to predict a concatenated sequence of variable number of keyphrases. Chan et al. (2019a) used reinforcement learning to enhance the task performance, whereas Swaminathan et al. (2020) performed preliminary studies on the use of Generative Adversarial Networks on the task. Chen et al. (2020) introduced a new decoding architecture (exclusive hierarchical decoding) to capture the hierarchical structure of keyphrases (ExHiRD). We use ExHiRD as one of our models for our analysis along with a transformer-based model (T5). Wu et al. (2021) take a joint training approach to learn both keyphrase extraction and generation through different layers instead of using a single seq2seq framework for both present and absent keyphrase prediction. Ye et al. (2021) decode multiple keyphrases in parallel while also using an assignment algorithm to reduce penalization from misaligned orders in predicted and gold keyphrases.

There have been a few empirical analyses on some aspects of the generation models. Meng et al. (2021) showed the experimental results for different hyperparameter changes including the change of ordering format for concatenating target keyphrases. Çano and Bojar (2019) explored the application of abstractive summarization techniques and evaluation metrics for keyphrase generation.

Calibration and uncertainty of neural models (Guo et al., 2017) have started to gain attention on several natural language processing tasks, including neural machine translation (Müller et al., 2019; Kumar and Sarawagi, 2019; Wang et al., 2020), natural language understanding (Desai and Durrett, 2020), and coreference resolution (Nguyen and O'Connor, 2015). For example, Wang et al. (2020) focused on the calibration of neural machine translation (NMT) models to understand the generative capability of the models at inference (decoding time) under the *exposure bias* (Ranzato et al., 2016), that captures the difference in training and inference caused by teacher forcing in autoregressive models. We explore the calibration of keyphrase generation models to better understand model behavior in this scenario.

3 Methodology

In this section, first, we briefly describe the two models: ExHiRD and T5; second, we formulate and define *keyphrase perplexity* and discuss calibration of generative models; lastly, we present a novel framework for soft-scoring-based evaluation of two sets of keyphrases.

3.1 Models

For our analysis, we consider two models: ExHiRD and T5. We chose ExHiRD because it is one of the strongest performing keyphrase generation architectures without relying on reinforcement learning or GAN. We chose T5 because applications of pre-trained Transformer-based models like T5 are becoming almost ubiquitous in NLP and T5 serves as a natural choice for keyphrase generation given its seq2seq architecture. Both models are trained on concatenated sequence of target keyphrases as in Yuan et al. (2020). Implementation details for the models are presented in Appendix A.

ExHiRD ExHiRD (Chen et al., 2020) is an RNN-based seq2seq model with attention and copy-mechanism. It uses a hierarchical decoding strategy to address the hierarchical nature of a sequence of keyphrases, where each keyphrase is, in turn, a sub-sequence of words. ExHiRD also proposes exclusion mechanisms to improve the diversity of keyphrases generated and reduce duplication.

T5 T5 (Raffel et al., 2020) is a pre-trained seq2seq Transformer (Vaswani et al., 2017), which is pre-trained on C4 corpus (a dataset with clean English text obtained by scraping the web). The T5 architecture includes an encoder-decoder architecture with various layers of self-attention and cross attention. We use `t5-base` model with 12 layers from the Transformers library (Wolf et al., 2020).

3.2 Understanding Model Behavior

3.2.1 Keyphrase Perplexity

We introduce the *Keyphrase Perplexity* metric to gauge model confidence on a particular predicted keyphrase. Keyphrase perplexity is rooted in the general concept of perplexity. Perplexity is a widely used metric for evaluating language models. For a sequence of tokens $w_{1:n} = w_1, w_2, ..., w_n$, of length n, perplexity is the inverse normalized probability p of generating them and can be defined as: $PP(w_{1:n}) = p(w_1, w_2, ..., w_n)^{-1/n}$. For an auto-regressive decoder, the probability p of the sequence can be factorized and reformulated as:

$$PP(w_{1:n}) = \left(\prod_{i=1}^{n} p(w_i | w_1, w_2, \ldots w_{i-1}) \right)^{-1/n} \tag{1}$$

We adapt this formulation to define keyphrase perplexity (KPP) over a sub-sequence $w_{j:k} = w_j, w_{j+1}, ..., w_k$ within the sequence $w_{1:n}$ ($1 \leq j \leq k \leq n$). Here, we assume that sub-sequence $w_{j:k}$ corresponds to a keyphrase. Our definition of $KPP(w_{j:k})$ is as follows:

$$KPP(w_{j:k}) = \left(\prod_{i=j}^{k} p(w_i | w_1, w_2, \ldots w_{i-1}) \right)^{-1/m} \tag{2}$$

where $m = k - j + 1$ is the number of tokens in the keyphrase $w_{j:k}$. Essentially, for KPP, we simply use the conditional probabilities of tokens within the keyphrase $w_{j:k}$ under consideration. During our analysis, any probability of the form $p(w_i | w_1, w_2, \ldots w_{i-1})$ indicates the predicted model probability for token w_i given that tokens $w_1, w_2, \ldots w_{i-1}$ have been already generated. We do not include starting, ending, separator, end of sequence tokens probabilities. As in perplexity, a lower keyphrase perplexity (KPP) indicates higher confidence in the prediction, whereas a higher KPP indicates lower confidence.

One limitation of this KPP formulation is that it does not negate the conditioning effect of previous keyphrases (included in sub-sequence w_1 to w_{j-1} while measuring the KPP of the keyphrase starting from w_j). However, removing this limitation is not straight-forward; so we take a naive assumption of treating the overall probabilities of keyphrases as independent of the other keyphrases.

3.2.2 Calibration of Generative Models

Model calibration includes modeling the accuracy of model predictions as a function of its generated posterior probabilities. A calibrated model has alignment between its empirical likelihood (accuracy) and its probability estimates. For example, a calibrated model that has a confidence of 90% while making predictions, would correctly predict 90 out of 100 possible samples. Formally, calibration models the joint distribution $P(Q, Y)$ over generated model probabilities $Q \in \mathbb{R}$ and labels Y. $P(Y = y | Q = q) = q$ signifies perfect calibration of a model (Guo et al., 2017).

Expected calibration error (ECE) is a popular measure of model miscalibration (Naeini et al., 2015). ECE is computed by partitioning the predictions according to their generated probabilites into k bins and summing up the weighted average of the absolute value of the difference between the accuracy and model confidence of a particular bin.

$$ECE = \sum_{i=1}^{k} \frac{b_i}{n} |acc(b_i) - confid(b_i)| \tag{3}$$

where n is the number of samples, b_i is the number of samples in the i^{th} bin with k bins, $1 \leq i \leq k$.

We also make use of reliability diagrams that depict the accuracy of the model as a function of the probability accross k bins. In Equation 2, we use KPP to gauge prediction perplexity by computing the inverse of the normalized value of the product of posterior probabilities for the tokens of a generated keyphrase. To bin keyphrases according to their posterior probabilities, we use inverse of KPP to plot the reliability diagrams and compute

ECE. Hence, the normalized posterior probability of a keyphrase is $(KPP)^{-1}$.

3.3 Soft Keyphrase Score (SoftKeyScore)

Previous work has mostly used extensions of standard F_1-based metrics to measure the performance of keyphrase generation models. Such evaluation metrics usually operate based on exact matches between predicted and gold keyphrases. Such a strategy cannot account for partial matches or semantic similarity. For example, if the prediction is "summarization model" and the gold is "summarization system", despite both semantic similarity and partial matching, the score will be 0. These kind of minor deviations are ubiquitous in keyphrase generation yet they are harshly penalized by the "exact match" evaluation metrics. We discuss more such examples in §4. This phenomenon motivates us to explore soft-scoring based evaluation metrics.

Çano and Bojar (2019) explored the use of metrics such as ROUGE that can accomodate for some level of partial matches but they are still suited mainly for comparing a sequence against another sequence. We want to compare a *set* of phrases with another *set*. Chan et al. (2019b) use Wikipedia information to control some level of name-variation over keyphrases of the same meaning but they still rely on strict binary scoring. In contrast to the above methods, we propose the SoftKeyScore as a suitable metric for evaluation between *sets* of sequences (keyphrases) as opposed to fully ordered sequences. We present our methodology below.

Assume we have two sets $G = \{g_1, g_2, ..., g_{|G|}\}$ and $P = \{p_1, p_2, ..., p_{|P|}\}$. G can be the set of gold keyphrases and P can be the set of predicted keyphrases. Assume we also have some soft-scoring function $score(x, y)$ which takes two phrases (x and y) as input and outputs a scalar $\in [0, 1]$ to indicate the degree of match between x and y. Given these elements, we propose the following evaluation framework:

$$P_{score} = \frac{1}{|P|} \cdot \sum_{p_i \in p} \max_{g_j \in G} score(p_i, g_j) \tag{4}$$

$$R_{score} = \frac{1}{|G|} \cdot \sum_{g_j \in g} \max_{p_i \in P} score(p_i, g_j) \tag{5}$$

$$F_{score} = 2 \cdot \frac{\cdot P_{score} \cdot R_{score}}{P_{score} + R_{score}} \tag{6}$$

Here, F_{score} indicates the final result of SoftKeyScore. It is analogous to F_1; the difference is how the precision and recall is computed. P_{score} and R_{score} are analogous to precision and recall, respectively. With a soft scoring function ($score$), however, one phrase p_i in set P can match with multiple phrases in set G. Thus, in Eqs. 4 and 5, we use a greedy matching strategy where we choose the maximum matching score for any comparison between a phrase in one set to all phrases in the other set. This overall framework is very similar to the framework used for BERTScore (Zhang et al., 2019). However, the crucial difference is that we are using a generic matching function to measure similarity between two *sequences* (keyphrases) instead of two token embeddings. In fact, one of our proposed scoring functions (discussed below) uses the BERTScore.

SoftKeyScore is invariant to the order of phrases. This is suitable in our context of evaluating *sets* of keyphrases. At the same time, by using the right $score$ function (like BERTScore), we can account for the order among the words within phrases (due to its contextualized embeddings). More on the implementation details of this framework can be found in Appendix B. Below we discuss two concrete instances of the $score$ function that we explore in our calculation of SoftKeyScore: Keyphrase Match Rate (KMR) score and BERTScore.

3.3.1 Keyphrase Match Rate (KMR)

We propose Keyphrase Match Rate (KMR) as the complement of Translation Error Rate (TER) (Snover et al., 2006). TER is used to evaluate predictions of neural machine translation (NMT) models by computing the number of edits required to modify the generated sequence into the target sequence. We slightly modify the original TER score by adding pad tokens to the shorter sequence (keyphrase) to keep the lengths of the two sequences in comparison equal. Pad tokens change some deletions to substitutions but that does not change the total edit cost since both have the same cost. This strategy ensures that TER stays in $[0, 1]$. Given that we want to measure the similarity between two keyphrases, we formulate KMR as: $1 - TER$. Given our modified TER, KMR also ranges in $[0, 1]$. A KMR score of 1 denotes a perfect match. KMR can account for the degree of partial matching between the two phrases although it can be deficient in capturing deeper aspects of semantic similarities.

Model	Inspec		Krapivin		SemEval		KP20k	
	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$
Present Keyphrases								
ExHiRD†	0.291	0.253	0.347	0.286	0.335	0.284	0.374	0.311
T5	0.340	0.287	0.328	0.271	0.306	0.275	0.387	0.335
Absent Keyphrases								
ExHiRD†	0.022	0.011	0.043	0.022	0.025	0.017	0.032	0.016
T5	0.025	0.014	0.053	0.028	0.023	0.016	0.036	0.018

Table 1: † indicates that the results are taken from Chen et al. (2020) but we used their publicly available code to reproduce the results. $F_1@5$ only keeps the top 5 keyphrase predictions (following Chen et al. (2020), incorrect keyphrases were added if there were < 5 predictions). $F_1@M$ uses the full model prediction for evaluation.

3.3.2 BERTScore

BERTScore (Zhang et al., 2019) is a recently proposed evaluation metric for evaluation of natural language generation models. BERTScore uses a similar method as described in Eqs. 4 to 6 but with the following differences:

1. Instead of sets (P and G) the evaluation is done on two sequences of tokens (prediction sequence and reference sequence).
2. Instead of phrases from some given sets, the equivalent of $score$ function in BERTScore compares contextualized token embeddings from the given sequences using dot-product.

In our context, we use BERTScore as another instance of the $score$ function as described previously to measure the similarity between two phrases. BERTScore can take into account both partial matching and deeper semantic similarities between the two phrases. Note that if we *just* use BERTScore replacing SoftKeyScore, the evaluation will no longer be invariant to the order of the keyphrases because of the use of contextualized embeddings over a “sequence" (it will no longer remain a set) of keyphrases.

4 Experiments and Results

4.1 Datasets

We select four widely used benchmarks for our experimentation: **KP20k** (Meng et al., 2017), **Krapivin** (Krapivin et al., 2009), **Inspec** (Hulth, 2003) and **SemEval** (Kim et al., 2010). We use KP20k training set (~500,000 samples) for training our models. We use KP20k test set and rest of the datasets (the test subset) for performance evaluation and analysis. Further implementation details are in Appendix A.

4.2 F_1 Evaluation Details

We used similar post-processing for evaluation as Chen et al. (2020). Concretely, we stemmed both target keyphrases and predicted keyphrases using Porter stemmer. We removed all duplicates from predictions after stemming. We determined whether a keyphrase is present or not by checking the stemmed version of the source document. For $F_1@5$, following Chen et al. (2020), if there were less than 5 predictions, we append incorrect keyphrases to the predictions to make it exactly 5.

4.3 Results, Analyses, and Observations

Model Performance (Exact match): We compare the results of T5 and ExHiRD using macro-averaged $F_1@5$ and $F_1@M$ metrics in Table 1. We find that despite lacking the advantage of pre-training, ExHiRD performs competitively with T5. Note that $F_1@M$ compares all the generated keyphrases with the gold labels whereas $F_1@5$ compares the first five keyphrases with the labels.

Keyphrase Perplexity Analysis: We compare keyphrase perplexities (KPP) of both T5 and ExHiRD. As can be seen from Figure 1, both models have lower KPP (thus, higher confidence) for present keyphrases than absent keyphrases. However, T5 is substantially more confident about its present keyphrase predictions compared to ExHiRD. This could be the effect of its pre-training. Both models tend to have higher KPP for absent keyphrases showcasing that they are having difficulty in learning to generate absent keyphrases.

In Figure 2, we show that the *conditional probabilities of tokens in a keyphrase* tend to be low at the boundaries (at the beginning of a keyphrase), but start to increase monotonically as the decoder

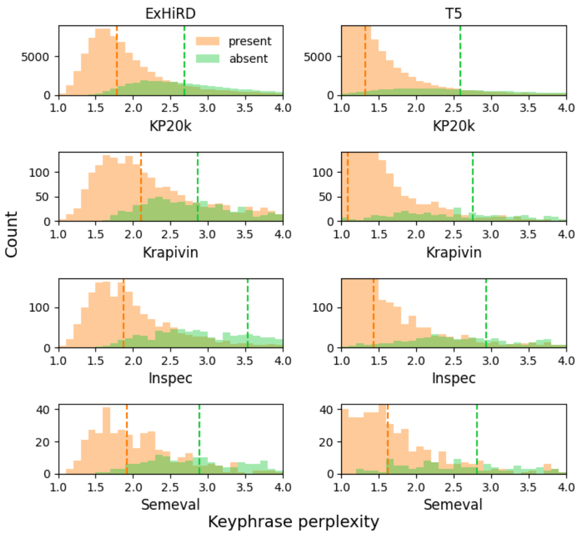

Figure 1: Histograms depicting number of keyphrases in keyphrase perplexity bins of size 0.1 for present and absent keyphrase generation. Dashed lines indicate the median of each distribution.

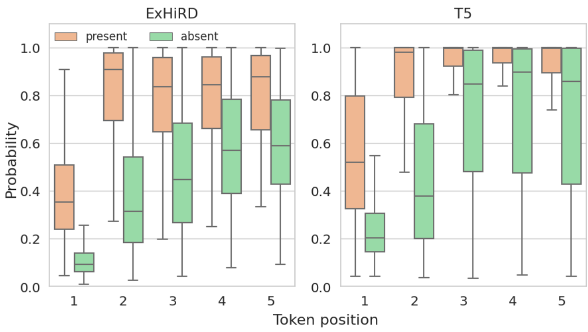

Figure 2: ExHiRD and T5's conditional probabilities for the first five tokens generated in a keyphrase (present and absent) in accordance to their relative positions within the keyphrase on KP20K test set. ExHiRD generates tokens at word-level; T5 generates at subword-level.

move towards the end of the keyphrase. Intuitively, it makes sense that a model will have less confidence predicting the start of a keyphrase because it requires settling on a specific keyphrase to generate out of many potential candidates. However, the first keyphrase token, once already generated, will condition and restrict the space of plausible candidates for the second token thereby increasing its confidence. For the same reason, probabilities near the end of a keyphrase tend to be much higher.

Model Calibration In Figure 1, we saw that T5 predicts keyphrases with higher model confidence than ExHiRD. But does the higher confidence actually translate into better predictions? Figure 3 shows the reliability diagrams for ExHiRD and T5 for both present and absent keyphrases. We can see that calibration of ExHiRD is better than T5. T5's high confidence keyphrase predictions does not translate into optimal accuracy values. In Table 2, T5's ECE is much higher than ExHiRD for all four datasets. We can say that T5 is an *overconfident* model. This may be due to the fact that T5 operates at subword level, and once the initial tokens of the keyphrase are predicted, T5 generates the rest of the tokens with very high confidence.

Dataset	ExHiRD	T5
Inspec	9.99	26.75
Krapivin	9.11	58.86
SemEval	10.18	26.64
KP20k	13.32	36.97

Table 2: Expected calibration error (ECE) for ExHiRD and T5 on various datasets. T5's calibration is worse than ExHiRD (lower the better).

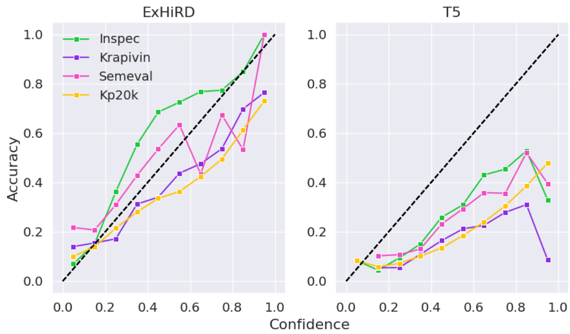

Figure 3: Reliability diagrams for model calibration of ExHiRD and T5. Dotted black line depicts perfectly calibrated model. We can see that ExHiRD is better calibrated than T5.

Dataset	Positional range 1	2	3	4	5
Inspec	1,326	845	686	602	173
Krapivin	706	206	182	159	59
SemEval	346	126	103	54	20
KP20k	39,571	9,865	8,313	6,317	1,704

Table 3: Number of keyphrases present keyphrases in gold labels binned into five sections, each having 20% characters of the source document.

Positional Variance We analyze both ExHiRD and T5's present keyphrase predictions with respect to their position in the input text. We divided the input text into five sections with 20% of characters in each, and binned the keyphrases appearing in them accordingly. In Table 3, we see that the majority of gold labels for the present keyphrases are in the first section (bin) of the input sequence. In Figure 4, we see that ExHiRD progressively fails to

Examples	F_1	F_{KMR}	$F_{BERTScore}$		
			DeBERTa	RoBERTA	SciBERT
Pred: performance evaluation, information retrieval, web search engine **Gold**: performance, information retrieval, world wide web, search engine	0.286	0.375	0.520	0.568	0.618
Pred: bgp, network engineering, routing protocols **Gold**: routing, traffic engineering, modeling, bgp	0.286	0.500	0.538	0.549	0.671
Pred: pwarx identification, chiu's clustering algorithm, affine sub model estimation, hyperplane partitions **Gold**: experimental validation, clustering, identification, hybrid systems, pwarx models, chiu's clustering technique	0.000	0.083	0.234	0.260	0.493

Table 4: Examples of F_{KMR} and $F_{BERTScore}$ with different pre-trained weights when compared against F1. F_{KMR} and $F_{BERTScore}$ indicates SoftKeyScore using KMR and BERTScore respectively

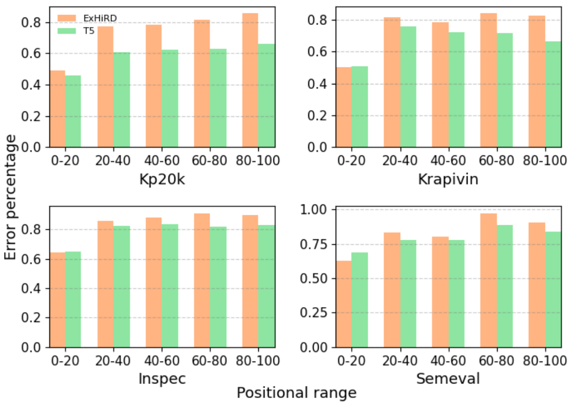

Figure 4: Error percentage of present keyphrase generation with respect to their position in the original text.

identify keyphrases in the later sections of the input text, whereas T5 not only performs well in identifying keyphrases present in the initial sections of the text, but it also performs better than ExHiRD in predicting keyphrases from the later sections (bins). This pattern is particularly prominent on KP20k. The bias towards predicting earlier present keyphrases is, most likely, further compounded by the fact that the present keyphrases are ordered according to their position of first occurrence within the target sequence. For an autoregressive model, it would be also likely to be easier to learn to predict the earlier sections of the target. As such the models can be biased to be good at only predicting keyphrases that occur early in the source text. However, the potential main reason for the bias is simply the fact that the majority of keyphrases exist in the earlier segments of a document as shown in Table 3. Nevertheless, T5 appears more resistant to these biases, despite being exposed to the same data and similarly ordered target sequences. These results hint also to a "better understanding" of the overall semantics of the document by the T5 model, and hence, its improved generation of short phrase document summaries (i.e., keyphrases).

SoftKeyScore Evaluation: Table 4 provides some concrete examples that demonstrate the potential of SoftKeyScore over standard F_1 measures. As we can see, F_1 metrics are quite low despite high similarities of the predictions and targets. SoftKeyScore, (with BERTScore), can better fit our intuitions about similarity between sets of phrases.

In Table 5, we experiment with various pre-trained transformer language models to compute BERTScore for SoftKeyScore. We use DeBERTa (He et al., 2021), RoBERTa (Liu et al., 2020), and SciBERT (Beltagy et al., 2019) to compute BERTScore. Further details about the models are in Appendix B. Overall, we see that SoftKeyScores over KMR and BERTScore have significantly more number of matches with partial or similar keyphrases when compared to baseline $F1@M$ scores in Table 1. This finding is particularly important when evaluating absent keyphrases. Using exact-match based F1, absent keyphrase performance is often too low to meaningfully compare. Some past work (Meng et al., 2017; Yuan et al., 2020) have even attempted to show just the recall after over-generation ($Recall@50$) of keyphrases. Such metrics can fail to capture the performance of the models in a more practical context. However, with SoftKeyScore we find much higher absent keyphrase performance (without being recall-oriented) allowing for more score-readability and better comparison. Interestingly, we find that ExHiRD is often outperforming T5 in SoftKeyScore compared to the hard (exact-match) F_1 evaluation.

Human evaluation To assess the quality of predicted keyphrases we use help from a CS majoring student. The student was asked to provide an appropriate score to signify the closeness between the predicted set of keyphrases and the gold set of keyphrases in $[0, 1]$. The student scored T5 prediction and the corresponding gold sets of 500 sample

Score	ExHiRD				T5			
	Inspec	Krapivin	Semeval	KP20k	Inspec	Krapivin	Semeval	KP20k
Present keyphrases								
F_{KMR}	0.366	0.366	0.393	0.408	0.392	0.347	0.349	0.415
F_{BS} DeBERTa	0.388	0.370	0.396	0.428	0.405	0.344	0.359	0.433
F_{BS} RoBERTa	0.442	0.434	0.467	0.459	0.459	0.414	0.464	0.466
F_{BS} SciBERT	0.588	0.572	0.528	0.588	0.587	0.550	0.490	0.589
Absent keyphrases								
F_{KMR}	0.042	0.076	0.042	0.054	0.049	0.071	0.040	0.054
F_{BS} DeBERTa	0.049	0.088	0.044	0.065	0.067	0.081	0.042	0.067
F_{BS} RoBERTa	0.072	0.135	0.087	0.083	0.089	0.122	0.086	0.087
F_{BS} SciBERT	0.160	0.253	0.128	0.173	0.187	0.212	0.117	0.182

Table 5: SoftKeyScore of present and absent keyphrase performance using KMR and BERTScore with different pre-trained weights. F_{KMR} and $F_{BERTScore}$ (F_{BS}) indicates SoftKeyScore with KMR and BERTScore respectively.

Metric	Metric ↔ Human
F_1	0.3664
F_{KMR}	0.4033
F_{BS} DeBERTa	0.3910
F_{BS} RoBERTa	0.3854
F_{BS} SciBERT	0.3543

Table 6: Pearson correlation for various metrics against human scores of sets of predicted and gold keyphrases.

documents from the KP20k test dataset.

In Table 6, we show the Pearson correlation between various metrics when compared against the human scores. We see that F_{KMR}, F_{BS} DeBERTa and F_{BS} RoBERTa are better correlated with human scores than the F1 metric. Interestingly, F_{BS} SciBERT has the worst correlation. We find that SciBERT is generally more generous (overly-optimistic) with the magnitude of its similarity score than the other metrics whereas the human judgment is on a more conservative (realistic) side. Thus, SciBERT did not align well with the human evaluation. F_{KMR}, which is generally more conservative in its scoring, has the best correlation with the human evaluation. However, F_1 is too conservative because even a minor difference in two keyphrases (predicted and gold) would imply a match score of 0 between them for F1.

5 Conclusion and Discussion

In this work, we evaluate, analyze, and compare two powerful seq2seq models for keyphrase generation—one is an RNN-based model (ExHiRD) with a hierarchical decoding strategy and another is a massively pre-trained Transformer-based model (T5). Moreover, we propose a novel and more powerful technique (SoftKeyScore) for evaluating keyphrase generation performance (using soft-matching instead of exact matching).

Findings and Future Directions Here, we discuss our main findings of the paper and motivate their use for future work. First, we find that the model confidence of absent keyphrase predictions are much lower than present keyphrase predictions for both models. Thus, the models know to be more uncertain with absent keyphrase generation (for which both models indeed have poor performance). However, upon checking for model calibrations, interestingly, we find that T5 is more overconfident (poorly calibrated) compared to ExHiRD. There is potential for further work on models' calibration.

Second, we find that the models are much less confident in predicting the starting tokens of a keyphrase. We believe deciding on the start of the keyphrase is much harder than predicting the follow-up tokens. Based on this finding, we may be able to make more efficient semi-autoregressive models that sequentially decode different keyphrases but simultaneously decode different tokens within a particular keyphrase.

Third, T5 is better at predicting present keyphrases from later positions in the given texts. This finding suggests that T5 may generalize better on out of domain datasets (e.g., legal documents) where there may be no strong bias for keyphrases to occur mainly in the early sections of documents. There is also room for extensions for better prediction of present keyphrases at later positions.

Fourth, we motivate and propose a soft-scoring based evaluation metric (SoftKeyScore) which we believe shows more potential than the standard F1-based metric. Particularly, absent keyphrase generation may gain more significant benefit from SoftKeyScore because generated abstractive keyphrases which are semantically similar (but non-identical at the lexical level) to a target keyphrase can be more meaningfully evaluated.

References

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. 2019. Memory efficient adaptive optimization. In *Advances in Neural Information Processing Systems*, volume 32, pages 9749–9758. Curran Associates, Inc.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3606–3611.

Florian Boudin, Ygor Gallina, and Akiko Aizawa. 2020. Keyphrase generation for scientific document retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1126.

Erion Çano and Ondřej Bojar. 2019. Keyphrase generation: A text summarization struggle. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 666–672, Minneapolis, Minnesota. Association for Computational Linguistics.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019a. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019b. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174, Florence, Italy. Association for Computational Linguistics.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *EMNLP*, pages 4057–4066.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. Title-guided encoding for keyphrase generation. In *AAAI*, pages 6268–6275.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK 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.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.

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

Steve Jones and Mark S Staveley. 1999. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Rui Meng, Xingdi Yuan, Tong Wang, Peter Brusilovsky, Adam Trischler, and Daqing He. 2019. Does order matter? an empirical study on generating multiple keyphrases as a sequence. *arXiv preprint arXiv:1909.03590*.

Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. 2021. An empirical study on neural keyphrase generation.

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.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2901–2907. AAAI Press.

Khanh Nguyen and Brendan O'Connor. 2015. Posterior calibration and exploratory analysis for natural language processing models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1587–1598, Lisbon, Portugal. Association for Computational Linguistics.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200. Citeseer.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3070–3079, Online. Association for Computational Linguistics.

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

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

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2Set: Generating diverse keyphrases as a set. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4598–4608, Online. Association for Computational Linguistics.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Implementation Details

ExHiRD is trained from the publicly available code [2] using the original settings mentioned in the paper (Chen et al., 2020). T5 was trained with SM3 optimizer (Anil et al., 2019) for its memory efficiency. We use a learning rate (lr) of 0.1 and a warm up for 2000 steps with the following formulation: $lr = lr{\cdot}minimum\left(1, \left(\frac{steps}{warmup\_steps}\right)^2\right)$ The learning rate was tuned among the following choices: $[1.0, 0.1, 0.01, 0.001]$. We use an effective batch size of 64 based on gradient accumulation. We train T5 for 10 epochs with a maximum gradient norm of 5. Both models were trained using teacher forcing. We use train, validation and test splits from Meng et al. (2017). Following (Meng et al., 2019; Chen et al., 2020), the keyphrases in the target sequence are ordered according to their position of first occurrence within the source text. The first occurring keyphrase in the source text appears first in the target sequence. The absent keyphrases were appended in the end according to their original order. Both T5 and ExHiRD experienced target sequences in that order during training. Predictions for both the models were generated through greedy decoding. We use a maximum length of 50 tokens for T5 during decoding.

We use a single NVIDIA V100 GPU for training and testing all our models.

B SoftKeyScore Implementation

When we use KMR, we first stem the phrases being compared with Porter Stemmer. We use the BERTScore implementation provided by the authors [3]. We use variations of pre-trained transformer model weights to compute BERTScore such as `microsoft/deberta-large-mnli` for DeBERTa, `roberta-large` for RoBERTa and `scibert-scivocab-uncased` for SciBERT. All the weights are streamlined and made available by Wolf et al. (2020). We also use baseline rescaling of BERTScore as done by Zhang et al. (2019). For both BERTScore and KMR based scoring functions, also use a threshold t of 0.4 such that the output of the score function becomes 0 if it is $< t$. This makes prevent inflation of the overall score from low scoring matches.

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

[3] https://github.com/Tiiiger/bert_score