Keyphrase Generation with Fine-Grained Evaluation-Guided Reinforcement Learning

Yichao Luo[*], **Yige Xu**[*], **Jiacheng Ye, Xipeng Qiu, Qi Zhang**[†]
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
Songhu Road 2005, Shanghai, China
{ycluo18,ygxu18,yejc19,xpqiu,qz}@fudan.edu.cn

Abstract

Aiming to generate a set of keyphrases, Keyphrase Generation (KG) is a classical task for capturing the central idea from a given document. Based on Seq2Seq models, the previous reinforcement learning framework on KG tasks utilizes the evaluation metrics to further improve the well-trained neural models. However, these KG evaluation metrics such as $F_1@5$ and $F_1@M$ are only aware of the exact correctness of predictions on phrase-level and ignore the semantic similarities between similar predictions and targets, which inhibits the model from learning deep linguistic patterns. In response to this problem, we propose a new fine-grained evaluation metric to improve the RL framework, which considers different granularities: token-level F_1 score, edit distance, duplication, and prediction quantities. On the whole, the new framework includes two reward functions: the fine-grained evaluation score and the vanilla F_1 score. This framework helps the model identifying some partial match phrases which can be further optimized as the exact match ones. Experiments on KG benchmarks show that our proposed training framework outperforms the previous RL training frameworks among all evaluation scores. In addition, our method can effectively ease the synonym problem and generate a higher quality prediction. The source code is available at https://github.com/xuyige/FGRL4KG.

1 Introduction

Keyphrase Generation (KG) is a classical but challenging task in Natural Language Processing (NLP), which requires automatically generating a set of keyphrases. Keyphrases are short phrases that summarized the given document. Because of the condensed expression, keyphrases can be beneficial to various downstream tasks such as information retrieval (Jones and Staveley, 1999), opinion mining (Wilson et al., 2005; Berend, 2011), document clustering (Hulth and Megyesi, 2006), and text summarization (Wang and Cardie, 2013).

In recent years, end to end neural models have been widely-used in generating both present and absent keyphrases. Meng et al. (2017) introduced CopyRNN, which consists of an attentional encoder-decoder model (Luong et al., 2015) and a copy mechanism (Gu et al., 2016). After that, relevant works are mainly based on the sequence-to-sequence framework (Yuan et al., 2020; Chen et al., 2018, 2019). Meanwhile, $F_1@5$ (Meng et al., 2017) and $F_1@M$ (Yuan et al., 2020) are used for evaluating the model prediction. $F_1@5$ computes the F_1 score with the first five predicted phrases (if the number of phrases is less than five, it will randomly append until there are five phrases). $F_1@M$ compares all keyphrases (variable number) predicted by the model with the ground truth to compute an F_1 score. Furthermore, Chan et al. (2019) utilize the evaluation scores as the reward function to further optimize the neural model throughout the reinforcement learning (RL) approach.

However, the traditional F_1-like metrics are on phrase-level, which can hardly recognize some partial match predictions. For example, supposing that there is a keyphrase called "*natural language processing*", and one model provides a prediction called "*natural language generation*" while another model provides "*apple tree*". Both of these two phrases will get zero score from either $F_1@5$ or $F_1@M$. But it is undoubtedly that "*natural language generation*" should be a better prediction than "*apple tree*". Chan et al. (2019) propose a method to evaluate similar words, but they only consider abbreviations of keywords and use it only during the evaluation stage.

In response to this problem, we propose a **F**ine-**G**rained (FG) evaluation score to distinguish these partial match predictions. First, in order to align the F_1 score, the exact correct predictions will ob-

[*] These two authors contributed equally.
[†] Corresponding author.

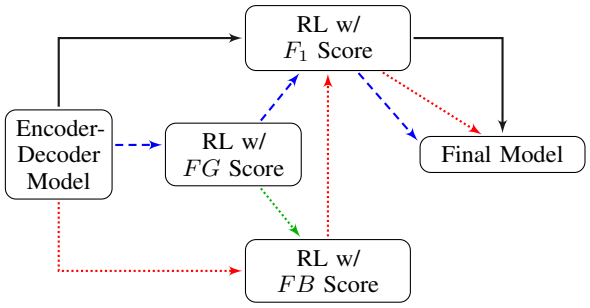


Figure 1: Flow chart of three reinforcement learning methods. The blue edges and red edges are our proposed reinforcement learning methods (catSeq*+2RL(FG) and catSeq*+2RL(FB)). The green densely dotted line means the FB score. We use some data generated by FG score to train the BERT model, and the BERT model is used to compute the FB score.

tain the FG score of one (e.g., *natural language processing* mentioned above), and the absolutely incorrect predictions will obtain the FG score of zero (e.g., *apple tree* mentioned above). Second, for partial match predictions like "*natural language generation*", FG score, our proposed metric, will compare the prediction with the target in the following perspectives:(1) prediction orders in token-level; (2) prediction qualities in token-level; (3) prediction diversity in instance-level; (4) prediction numbers in instance-level. The specific detail of our proposed FG score can be seen in Section 3.3.

Based on previous works that use the reinforcement learning technique and adopt the self-critical policy gradient method (Rennie et al., 2017), we propose a two-stage RL training framework for better utilizing the advantages of FG score. As shown in Figure 1, the black edges show the previous RL process and the blue edges show our proposed two-stage RL process. Our two-stage RL can be divided into two parts: (1) First, we set FG score as the adaptive RL reward and use RL technique to train the model; (2) Second, we use F_1 score as the reward, which is the same as Chan et al. (2019). Furthermore, in order to make FG score smoothly, we carefully train a BERT (Devlin et al., 2019) model to expand the original FG score from discrete to continuous numbers (the green line in Figure 1). This BERT scorer can predict a continuous FG score, which can also be used in our two-stage RL framework (the red edges in Figure 1).

Comparing with the F_1 score, our FG score has two main advantages: (1) FG score can recognize some partial match predictions, which can better evaluate the quality of predictions in a fine-grained dimension; (2) During the reinforcement learning stage, FG score can provide a positive reward to the model if it predicts some partial match predictions, while the F_1 score will return a negative reward of zero in this situation. Therefore, in our proposed two-stage RL framework, the first stage can help the model predict some partial match phrases, and the second stage can further promote the partial match phrases to the exact match phrases. We conduct exhaustive experiments on keyphrase generation benchmarks and the results show that our proposed method can help better generating keyphrases by improving both the traditional F_1 score and the FG score. In addition to this, we also conduct experiments to analyze the effectiveness of each module.

Our main contributions are summarized as follows:

- We propose FG score, a new fine-grained evaluate metric for better distinguish the predicted keyphrases.
- Base on our evaluation metric, we propose a two-stage reinforcement learning method to optimize the model throughout a better direction.
- We train a BERT-based scorer whose corpus come from previous training. The scorer can effectively perceive the similarity of two keyphrases on semantic level.
- We conduct exhaustive experiments and analysis to show the effectively of our proposed FG metric.

2 Related Work

In this section, we briefly introduce keyphrase generation models and evaluation metrics.

2.1 Keyphrase Generation Models

In KG task, keyphrases can be categorized into two types: *present* and *absent*, depending on whether it can be found in the source document or not. In recent years, end to end neural model has been widely-used in generating both present and absent keyphrases. Meng et al. (2017) introduced CopyRNN, which consists of an attentional encoder-decoder model (Luong et al., 2015) and a copy

mechanism (Gu et al., 2016). After that, relevant works are mainly based on the sequence-to-sequence framework. More recently, Chen et al. (2018) leverages the coverage (Tu et al., 2016) mechanism to incorporate the correlation among keyphrases, Chen et al. (2019) enrich the generating stage by utilizing title information, and Chen et al. (2020) proposed hierarchical decoding for better generating keyphrases. In addition, there are some works focus on keyphrase diversity (Ye et al., 2021), selections (Zhao et al., 2021), different module structure (Xu et al., 2021), or linguistic constraints (Zhao and Zhang, 2019).

2.2 Keyphrase Generation Metrics

Different to other generation tasks that need to generate long sequences, KG task only need to generate some short keyphrases, which means n-gram-based metrics (e.g., ROUGE (Lin, 2004), BLEU (Papineni et al., 2002)) may not suitable for evaluations. Therefore, $F_1@5$ (Meng et al., 2017) and $F_1@M$ (Yuan et al., 2020) are used to evaluate the keyphrases which is predicted by models. This evaluation score is also used as an adaptive reward to improve the performance through reinforcement learning approach (Chan et al., 2019).

3 Methodology

3.1 Problem Definition

In this section, we will briefly define the keyphrase generation problem. Given a source document $\mathbf{x}$, the objective is to predict a set of keyphrases $\mathcal{P} = \{p_1, p_2, \ldots, p_{|\mathcal{P}|}\}$ to maximum match the ground-truth keyphrases $\mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$, where $|\mathcal{P}|$ and $|\mathcal{Y}|$ are the number of the predicted keyphrases and the number of ground truth keyphrases respectively. Both source document $\mathbf{x} = [x_1, ..., x_{|\mathbf{x}|}]$ and a keyphrase in the set of target keyphrases $y_i = [y_{i,1}, ..., y_{i,|y_i|}]$ are words sequences, where $|\mathbf{x}|$ and $|y_i|$ represent the length of source sequence $\mathbf{x}$ and the i-th keyphrase sequence $\mathbf{y}_i$, respectively.

3.2 Seq2Seq Model with Minimizing Negative Log Likelihood Training

In this section, we descibe the Seq2Seq model with attention (Luong et al., 2015) and copy mechanism (Gu et al., 2016), which is our backbone model.

Encoder-Decoder Model with Attention We first convert the source document $\mathbf{x} = [x_1, x_2, ..., x_{|\mathbf{x}|}]$ to continuous embedding vectors $\mathbf{e} = [e_1, e_2, ..., e_{|\mathbf{x}|}]$. Then we adopt a bi-directional Gated-Recurrent Unit (GRU) (Cho et al., 2014) as the encoder to obtain the hidden state $\mathbf{H} = \text{Encoder}(\mathbf{e})$.

Then another GRU is adopted as the decoder. At the step t, we compute the decoding hidden state $\mathbf{S}_t$ as folow:

$$\mathbf{S}_t = \text{Decoder}(\mathbf{e}_{t-1}, \mathbf{s}_{t-1}) \tag{1}$$

In addition, we incorporate the attention mechanism (Luong et al., 2015) to compute the contextual vector $\mathbf{u}$ which represents the whole source document at step t:

$$\mathbf{u}_t = \sum_{j=1}^{T} \alpha_{tj} \mathbf{H}_j \tag{2}$$

where α_{tj} represents the correlation between the source document at position j and the output of the decoder at step t.

Copy Mechanism Because there are a certain number rare words in the document, traditional Seq2Seq models perform pooly when predicting these rare words. Thus, we introduce the copy mechanism (Gu et al., 2016) to alleviate the out-of-vocabulary (OOV) problem. The probability of producing a token contains two parts: the probability for generation p_g and probability for copy mechanism p_c. p_g is estimated by a standard language model based on the global vocabulary, and p_c is estimated by the copy distribution based on local vocabulary which only contain one case. The definition of p_c is:

$$p_c\left(y_{i,t}|y_{i,<t}, \mathbf{H}\right) = \frac{1}{Z} \sum_{j:x_j=y_{i,t}} e^{\omega(x_j)}, y_{i,t} \in \chi \tag{3}$$

where χ represents the set of all rare words in the source document and Z is used for normalization.

Minimizing Negative Log Likelihood Training Finally, we train all parameters in the model θ by minimizing the negative log likelihood loss:

$$\mathcal{L}(\theta) = -\sum \log P\left(y_{i,t}|y_{i,<t}, \mathbf{H}\right) \tag{4}$$

3.3 Fine-Grained Score

Because traditional KG methods only care predictions on phrase-level in evaluate stage, they ignore

both information on token-level and instance-level. Not only in the traditional Seq2Seq model, there are also in RL training. The environment also calculates the reward (recall or F_1-score) only in phrase-level, which ignores the overall performance of prediction. Due to this problem, the training process may go in the wrong direction (e.g. Model will give a zero score for a phrase that has more than half right). Thus, we propose a new metric: **F**ine **G**rained Score (FG), which considers both token-level and instance-level information. It is divided into the following four parts.

3.3.1 Phrase Similarity on Token Level

For the comprehensive calculation later, we first compute the similarity score between a predicted phrase and a ground-truth phrase on token-level. We use edit distance and token-level F_1 score as our metric.

Obviously, in order to get token-level similarity, Guaranteed the correctness of phrase on token-level is important. So token-level F_1 score is necessary. Given a predicted phrase $p_i \in \mathcal{P}$, we use $\mathbf{F}_1(p_i, y_j)$ represent the score for i-th predicted phrase and j-th ground-truth phrase.

Because the order of the words in a phrase is also important, we introduce the edit distance to measure the sequential difference between the two phrases. Particularly, the edit distance $\mathbf{ED}(p_i, y_j)$ denotes how many times should modify p_i to y_j at least, where one time only can modify one word and modify operation only contains three operations: **add**, **delete**, **change**. We use dynamic programming to calculate the edit distance as follow:

$$D_k^m = \begin{cases} \min(D_{k-1}^{m-1}, D_k^{m-1} + 1, D_{k-1}^m + 1) \\ \text{if } p_{i,k} = y_{j,m} \\ \\ \min(D_{k-1}^{m-1} + 1, D_k^{m-1} + 1, D_{k-1}^m + 1) \\ \text{if } p_{i,k} \neq y_{j,m} \end{cases} \tag{5}$$

where D_k^m denotes minimum number of modifications for transforming first k token in p_i to first m token in y_j. $k \in [1, |p_i|]$ and $m \in [1, |y_j|]$. Because the more modifications there are, the less similar the two sequences are, the $\mathbf{ED}(p_i, y_j)$ score can be formulated as follow:

$$\mathbf{ED}(p_i, y_j) = 1 - \frac{D_{|p_i|}^{|y_j|}}{\max\{|p_i|, |y_j|\}} \tag{6}$$

And for a instance ($\mathbf{x}$, $\mathcal{Y}$, $\mathcal{P}$), we compute score list $\mathbf{scoreL}$ as follow:

$$\mathbf{scoreL}_i = \max_{y_j}\{\frac{\mathbf{ED}(p_i, y_j) + \mathbf{F}_1(p_i, y_j))}{2}\}, \tag{7}$$

where $\mathbf{F}_1$ is token-level F_1 score. $i \in [1, |\mathcal{P}|]$ and $j \in [1, |\mathcal{Y}|]$. And we use a maximum-match score to a particularly predicted phrase.

3.3.2 Global Generation Quality on Instance Level

In Section 3.3.1, we proposed a method to compute the phrase similarity on token level. In this section, we will further consider the generation quality on instance level.

There are many factors can influent the global generation quality, but we select the most representative factors: diversity and the prediction quantities. Therefore, we use a **Repetition Rate Penalty** and **Generation Quantity Penalty** for the FG score, which is shown in Algorithm 1.

Algorithm 1 Global Generation Quality Penalty

Input:
The set of ground-truth keyphrases, $\mathcal{Y}$;
The set of predicted keyphrases $\mathcal{P}$;
The score list of prediction, **scoreL**
Output:
reward for an instance;

```
// Repetition rate penalty
initial two dicts dictY and dictP
for all keyphrase y_i in Y do
    for all word y_ij in y_i do
        dictY[y_ij] = dictY[y_ij] + 1
reverse sort P and scoreL by key scoreL
for i = 0; i < |P|; i++ do
    for all word p_ij in p_i do
        if p_ij in dictY then
            dictP[p_ij] ++
            if dictP[p_ij] > dictY[p_ij] then
                scoreL[i] = 0
finalscore = sum(scoreL) / |P|
// Generatation quantity penalty
corr = 1.0 - (|Y|-|P|)^2 / max(|Y|,|P|)^2
finalscore = finalscore * corr
return finalscore
```

The first factor is the repetition rate penalty. This operation means that there is a punishment if the model predicts similar keyphrases greater equal

than twice, which can also reduce the duplication. We first count the words that appear in the ground truth. Then we sort the prediction and the score list in reverse according to the score. After that we iterate the prediction list and count the words that appear in the predictions. Once a word appears in the predictions twice more than that in the ground truth, we claim this token "repetitive". Based on this, the corresponding phrase is labelled as invaild. Lastly we will compute an average score of all phrases as a normalization, which can be used to represent the score of the corresponding instance.

The second factor is the generation quantities. The model will obtain the highest score if it predicts only one most simple phrase because it is an exact match result in most cases. Therefore, we add a generation coefficient to solve this problem.

3.4 Continuous Scorer

In fact, although the FG metric includes the token-level and instance-level information for keyphrase, deeper semantic information is not considered. As the introduction says, when "*natural language processing*" is ground truth, our FG metric will give "*natural language understanding*" and "*natural language generation*" a same score. But "*natural language understanding*" and "*natural language generation*" have different semantics. In order to solve this problem, we incorporate pre-train model (e.g., BERT) to train a continuous scorer which denotes the similarity of two keyphrases.

Because many tuples (p_i, y_j, $\mathbf{scoreL}_i$) are generated when we compute the FG score, we screen portions as training corpus for BERT. We concatenate the p_i and y_j as (`[CLS]` p_i `[SEP]` y_j `[SEP]`) to a sequence as input for BERT scorer, where `[CLS]` and `[SEP]` is the same as the vanilla BERT (Devlin et al., 2019). In the training stage, $\mathbf{scoreL}_i$ score is used as the supervised target.

After get the BERT scorer, we can easily evaluate the similarity of two keyphrase. Similar to the Eq (7), for a instance ($\mathbf{x}$, $\mathcal{Y}$, $\mathcal{P}$), we compute BERT-based score list $\mathbf{scoreLB}$ as follow:

$$\mathbf{scoreLB}_i = \max_{y_j}\{\mathrm{BERT}(p_i, y_j)\}. \quad (8)$$

where $i \in [1, \mathcal{P}]$ and $j \in [1, \mathcal{Y}]$. Finally, we also put $\mathbf{scoreLB}_i$ into Algorithm 1 to compute finally BERT-based score (also called FB score).

3.5 Reinforcement Learning

In this section, we will briefly describe our proposed two-stage reinforcement learning method.

3.5.1 Vanilla RL Training

Reinforcement learning has been widely applied to text generation tasks, such as machine translation (Wu et al., 2018), summarization (Narayan et al., 2018), because it can train the model towards a non-differentiable reward. Chan et al. (2019) incorporate reinforce algorithm to optimize the Seq2Seq model with an adaptive reward function. They formulate keyphrase generation as follow. At the time step $t = 1, \ldots, T$, the agent produces an action (token) $\widehat{y}_t$ sampled from the policy (language model) $P(\widehat{y}_t|\widehat{y}_{<t})$, where $\widehat{y}_{<t}$ represent the sequence generated before step t. After generated t-th tokens, the environment $\widehat{s}_t$ will gives a reward $r_t(\widehat{y}_{<=t}, \mathcal{Y})$ to the agent and updates the next step with a new state $\widehat{s}_{t+1} = (\widehat{y}_{<=t}, \mathbf{x}, \mathcal{Y})$. We repeat the above operations until generated all token. Typically, the recall score or the F_1 score are used as the reward function.

3.5.2 Two-Stage RL Training

In the vanilla RL training, the reward is polarized in the phrase level: one for an exact match prediction and zero for other situations, which means a partial match phrase receives the same reward as an exact mismatch phrase. In order to help to recognize these partial match phrases during the training stage, we propose a two-stage RL training method. In the first stage, we use our new metric (FG score or FB score) as a reward to train the model. Then we apply the vanilla RL (using F_1 score) training as the second training stage. The whole RL training technique is similar to Chan et al. (2019), while we re-write the reward function.

4 Experiment

4.1 Dataset

We evaluate our model on three public scientific KG dataset, including **Inspec** (Hulth and Megyesi, 2006), **Krapivin** (Krapivin et al., 2009), **KP20k** (Meng et al., 2017). Each case from these datasets consists of the title, abstract, and a set of keyphrases. Following the previous work (Chen et al., 2020), we concatenate the title and abstract as input document, and use the set of keyphrases as labels. The same as the previous works above, we use the largest dataset, **KP20k**, to train the model,

Model	Inspec			Krapivin			KP20k		
	$F_1@M$	$F_1@5$	FG	$F_1@M$	$F_1@5$	FG	$F_1@M$	$F_1@5$	FG
catSeq(Yuan et al., 2020)	0.262	0.225	0.381	0.354	0.269	0.352	0.367	0.291	0.371
catSeqD(Yuan et al., 2020)	0.263	0.219	0.385	0.349	0.264	0.350	0.363	0.285	0.369
catSeqCorr(Chen et al., 2018)	0.269	0.227	0.391	0.349	0.265	0.360	0.365	0.289	0.374
catSeqTG(Chen et al., 2019)	0.270	0.229	0.391	0.366	0.282	0.344	0.366	0.292	0.369
SenSeNet(Luo et al., 2020)	0.284	0.242	0.393	0.354	0.279	0.355	0.370	0.296	0.373
ExHiRD-h(Chen et al., 2020)	0.291	0.253	0.395	0.347	0.286	0.354	0.374	0.311	0.375
Utilizing RL (Chan et al., 2019)									
catSeq+RL(F_1)	0.300	0.250	0.382	0.362	0.287	0.360	0.383	0.310	0.369
catSeqD+RL(F_1)	0.292	0.242	0.380	0.360	0.282	0.357	0.379	0.305	0.377
catSeqCorr+RL(F_1)	0.291	0.240	0.392	0.369	0.286	0.376	0.382	0.308	0.377
catSeqTG+RL(F_1)	0.301	0.253	0.389	0.369	0.300	0.344	0.386	0.321	0.370
Ours									
catSeq*+RL(FG)	0.252	0.201	0.460	0.359	0.228	0.413	0.365	0.290	0.440
catSeq*+RL(FB)	0.254	0.200	**0.463**	0.354	0.230	**0.416**	0.366	0.291	**0.444**
catSeq*+2RL(FG)	0.308	0.266	0.425	**0.375**	0.304	0.389	0.391	0.327	0.381
catSeq*+2RL(FB)	**0.310**	**0.267**	0.430	0.374	**0.305**	0.390	**0.392**	**0.330**	0.383

Table 1: Result of present keyphrase prediction on three datasets. "RL" denotes that a model is trained by one-stage reinforcement training. "2RL" denotes that a model is trained by two-stage RL training. The notation in parentheses denotes the reward function in first RL training stage. All second reward function in two-stage RL training is F_1 score. "catSeq*" represents that we select the best model of four different catSeq-based baseline models. FB indicates that the reward is computed by the continuous BERT scorer. The underline numbers represent the best result in previous work. FG is the metric we propose.

and use all datasets to evaluate the performance of our model. After same data pre-processing as Chan et al. (2019), **KP20k** dataset contains 509,818 training samples, 20,000 validation samples, and 20,000 testing samples.

4.2 Evaluation Metrics

Most previous work (Meng et al., 2017; Chen et al., 2018, 2019) cutoff top k (which k is a fixed number) predicted keyphrases to calculate metrics such as $F_1@5$ and $F_1@10$. Due to the different number of keyphrases in different samples, Yuan et al. (2020) propose a new evaluation metric, $F_1@M$, which compares all keyphrases predicted with the ground-truth and compute the F_1 score. We evaluate the performance of our model using three different metrics, $F_1@5$, $F_1@M$, and FG (ours). After computing every samples' scores, we apply marco average to aggregate the evaluation scores. The same as Chan et al. (2019), we append random wrong keyphrases to prediction until it reaches five or more, because our method generates diverse keyphrases that usually less than five predictions.

4.3 Baseline Model

Following the name set of the previous works(Chan et al., 2019; Chen et al., 2020), we use four generative model trained under minimize the negative log likelihood loss, include **catSeq**(Yuan et al., 2020), **catSeqD**(Yuan et al., 2020), **catSeqCorr**(Chen et al., 2018), **catSeqTG**(Chen et al., 2019), **ExHiRD-h**(Chen et al., 2020). Because reinforcement learning is applied to our method, we also compare four reinforced model (Chan et al., 2019) include **catSeq+RL**, **catSeqD+RL**, **catSeqCorr+RL**, **catSeqTG+RL**. Each reinforced model is correspond to previous model applied RL approach. In this paper, our RL framework trains four models for comparison:

- **catSeq*+RL**(FG) and **catSeq*+RL**(FB) denotes that one-stage reinforcement learning training with FG-score reward or BERT-based reward.
- **catSeq*+2RL**(FG) and **catSeq*+2RL**(FB) denotes that two-stage RL training. Two methods use FG-score and BERT-based reward in first stage respectively, and both use F_1 reward in second score which is same as Chan et al. (2019).

5 Result and Analysis

5.1 Present Keyphrase Prediction

In this section, we evaluate the performance of our models on present keyphrase predictions using three different metrics, $F_1@M$, $F_1@5$, and FG, respectively. Table 1 shows the result of all baseline models and our proposed four models. From the result, we summarized our observations as follow:

Model	Phrase-level Result			Token-level Result		
	$F_1@M$	$F_1@5$	FG	tF	tP	tR
catSeq*+2RL(FG)	**0.391**	**0.330**	**0.383**	0.494	0.493	0.495
w/o ED	0.387	0.325	0.370	0.494	0.491	**0.498**
w/o TF	0.389	0.327	0.372	0.485	0.483	0.487
w/o RRP	0.390	0.328	0.375	**0.497**	**0.494**	0.500
w/o GNP	0.388	0.320	0.372	0.489	0.493	0.486

Table 2: Ablation study of catSeq*+2RL(FG) on **KP20k** dataset. ED means Edit Distance, TF means Token-level F_1 score (see Section 3.3.1), RRP means Repetition Rate Penalty, GNP means Generated Number Penalty (see Section 3.3.2). "w/o" means "without". tF, tP, tR means token-level metric.

(1) Our proposed methods achieve the state-of-the-art result on KG generation, which proves that it is necessary to deal with the semantic similarities between predictions and targets.

(2) In the phrase level, the reward returned by the vanilla RL method (with F_1 score) is polarized. Assuming that there are two partial match predictions in the baseline model (catSeq*), one of them may change into an exact match keyphrase while another may change into an exact mismatch keyphrase after the vanilla RL method. This phenomenon will increase the F_1 score, but only a similar FG score can be obtained. Therefore, the vanilla RL method hardly improves the quality of generation, although it improves the $F_1@5$ and $F_1@M$ score.

(3) We observe that the one-stage RL training (catSeq*+RL(FG)) induces the performance drop on both $F_1@M$ and $F_1@5$, especially on $F_1@5$, but it improves the performance on FG. The reason is that the number of predicted keyphrases is less than vanilla RL training. We predict 3.2 present keyphrases on average, and the vanilla RL training predicts 3.8 when ground truth is 3.3. We conclude that the number of our predictions is more reasonable comparing with the vanilla RL methods.

(4) Models with two-stage RL training far outperform those with only one-stage RL training on $F_1@M$ and $F_1@5$ metrics. Moreover, it shows that the vanilla RL training with F_1 score can effectively improve $F_1@5$ and $F_1@M$ after first stage training because first stage training improves the token-level quality of prediction.

(5) We observe that using BERT as a reward scorer makes the models perform better than using FG, indicating that the reward score produced by BERT is usually more accurate.

5.2 Ablation Study

To further examine the benefits that each component of the FG score brings to the performance, we conduct an ablation study on the **catSeq*+2RL(FG)** model. Our proposed methods are evaluated on the largest dataset **KP20k**. The results are shown in Table 2.

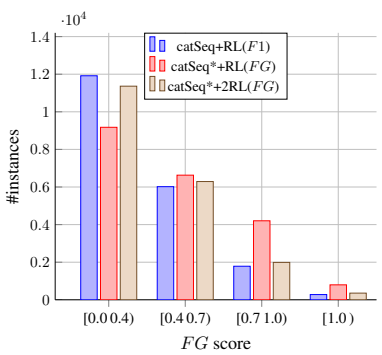
First, removal of edit distance score (w/o ED) does not affect model's performance on token-level but leads to performance drop most on phrase-level. Thus, it proves that edit distance is the most crucial in FG scores. Moreover, after we get rid of token-level F_1 (w/o TF), we observe that the phrase-level performance does not decrease much, but token-level performance decrease much. Therefore, we prove the effectiveness of token-level F_1 for token-level quality.

Compared with **catSeq*+2RL(FG)**, removal of the repetition rate penalty (w/o RRP) will cause the performance drop consistently on phrase-level, which indicates that RRP has a great effect on phrase-level $F_1@5$. Furthermore, for token-level results, we observe that the token-level recall and token-level F_1 score decreases somewhat, but token-level precision gets a promotion. From predicted results, we also obtain some observations when the lack of repetition rate penalty. There are a large number of keyphrase such as "*natural processing*", "*natural language*", "*natural natural natural*", when the ground-truth keyphrase is "*natural language processing*". The situation leads to high token-level accuracy but low overall performance.

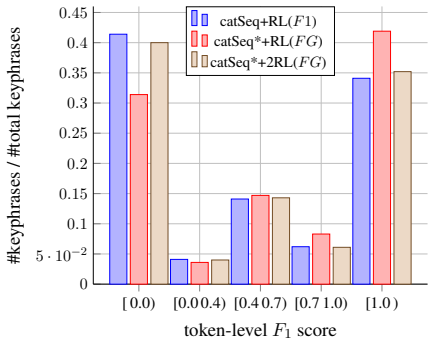
Finally, removal of generated number penalty (w/o GNP) will mainly cause the phrase-level $F_1@5$ to go down. We find that model tends to generate a small number of keyphrases as the predicted results because generating multiple keyphrases will reduce the reward. According to the definition of $F_1@5$, if the model can not generate enough five keyphrases, we should randomly add a mistake keyphrase to five. Thus, if we generate more keyphrases appropriately, $F_1@5$ will definitely get a boost. So in this situation, $F_1@5$ will decrease a lot. From what has been discussed above, all the

Document: Recent improvements in propositional satisfiability techniques (SAT) made it possible to tackle successfully some hard real-world problems (e.g. model-checking, circuit testing, propositional planning) by encoding into SAT. However, a purely boolean representation is not expressive enough for many other real-world applications, including the verification of timed and hybrid systems, of proof obligations in software, and of circuit design at RTL level. These problems can be naturally modeled as satisfiability in Linear Arithmetic Logic (LAL), i.e., ... We first investigate the relative benefits and drawbacks of each proposed technique by comparison with respect to a reference option setting. We then demonstrate the global effectiveness of our approach by a comparison with several state-of-the-art decision procedures. We show that the behavior of MATHSAT is often superior to its competitors, both on LAL, and in the subclass of Difference Logic.	
Ground-truth: propositional satisfiability; linear arithmetic logic; satisfiability module theory; integrated decision procedures	
catSeq+RL(F1) predictions: propositional satisfiability; linear regression; partition mathematical reasoning; propositional satisfiability experiment	FG = 0.333
catSeq+RL(FG) predictions: propositional satisfiability; linear arithmetic regression; boolean representation reasoning; mutiple decision trees	FG = 0.500
catSeq*+2RL(FG) predictions:propositional satisfiability; linear arithmetic logic; mathematical reasoning; mutiple decision procedures	FG = 0.667
catSeq*+2RL(FB) predictions:propositional satisfiability; linear arithmetic logic; integrated decision process; satisfiability problem	FG = 0.758

Figure 2: Case study for catSeq+RL(F_1), catSeq+RL(FG), catSeq*+2RL(FG) and catSeq*+2RL(FB). The red words represent the present keyphrases, the blue words represent the absent keyphrase. The green words represent the synonym with ground truth. The yellow words represent the duplicate part of a keyphrase. The underlined words represent correctly words on token-level.



(a) Number distribution histogram of FG score



(b) Proportion distribution histogram of token-level F_1 score

Figure 3: The number distribution of the FG and the proportion distribution of token-level F_1 score on test dataset by three different training process. catSeq+RL(F_1) denotes that one-stage RL training with F_1-score reward. catSeq*+RL(FG) denotes that one-stage RL training with FG-score reward. catSeq*+2RL(FG) denotes that two-stage RL training with FG and F_1-score reward.

modules in the FG score have their contribution.

5.3 Case Study

To better understand what benefits our proposed method brings, we present a case study with a document sample. As shown in the Figure 2, we compare the predictions generated by vanilla RL model (catSeq+RL(F_1)), two-stage RL training with FG score (catSeq*+2RL(FG)) and two-stage RL training with BERT scorer (catSeq*+2RL(FB)) on a same document sample. Overall, our two approaches have improved relative to the baseline model on FG scorer, and it shows that our overall generation quality has been improved.

From the case, we have three observations: **First**, catSeq*+2RL(FG) and catSeq*+2RL(FB) correctly predict the keyphrase "*linear arithmetic logic*" while catSeq+RL(F_1) predicts a "*linear regression*" which gets only one word right. It indicates that our two methods can improve the prediction quality on token-level and then finally improve the performance on phrase-level. **Second**, catSeq+RL(F_1) predicts two similarly keyphrases "*proposition satisfiability*" and "*proposition satisfiability experiment*", which our two methods do not. It fully demonstrates that our repetitive punishment plays an important role, which makes the predictions become diverse. **Third**, catSeq*+2RL(FB) generates a keyphrase "*integrated decision process*", which is synonym for ground truth "*integrated decision procedures*". It indicates that the BERT scorer can effectively perceive the semantics of keyphrases, which guides the training process of reinforcement learning.

5.4 Generative Quality Analysis

In this section, we analyze the prediction quality generated by vanilla RL model (catSeq+RL(F_1)), one-stage RL training with FG score (catSeq*+RL(FG)) and two-stage RL training with FG score (catSeq*+2RL(FG)) on instance-level and token-level respectively. We

both divided the FG score and token-level F_1 score into five parts. We conduct detailed analysis in the following.

In Figure 3a, we use the distribution of FG scores to analyze the generation quality on instance-level. By comparing the distribution of catSeq+RL(F_1) and catSeq*+RL(FG), we find that catSeq*+RL(F_1) has a larger proportion when FG score is low and catSeq*+RL(FG) has a larger proportion when FG score is high. It shows that using the FG score as a reward can improve the overall quality of predictions. Especially when score = 1.0 (which means all keyphrase is correctly in this instance), the number of catSeq*+RL(FG) is nearly three times as large as catSeq+RL(F_1). When comparing catSeq*+2RL(FG) with catSeq+RL(F_1), we obtain the similar conclusion as before. It is proved that the overall quality of the generated keyphrases can be improved after the first stage of reinforcement learning training.

In Figure 3b, due to different number keyphrases predicted by the model, we use the distribution of the proportion of the token-level F_1 score to analyze the generation quality on token-level. By comparing the distribution of catSeq+RL(F_1) and catSeq*+RL(FG), we find that catSeq*+RL(F_1) has a larger proportion when token-level F_1 is low and catSeq*+RL(FG) has a larger proportion when token-level F_1 is high. It indicates that the model can generate more keyphrases with more correct words throughout the reinforcement learning training with the FG score. (e.g. When groud truth is "*natural language processing*", catSeq+RL(F_1) generates "*natural X X*" and catSeq*+RL(FG) generates "*natural language X*". "*X*" means the inaccuracy word). This improvement also benefits to catSeq*+2RL(FG).

5.5 Human Evaluation for Continuous Scorer

As shown in Section 3.4, our continuous BERT-based scorer is an implicit and automatic. In this section we manually evaluate it to verify its effectiveness. We randomly selected 1000 pairs of matching predicted and ground-truth keyphrases in the training of reinforcement learning with BERT-based rewards and save a BERT score at the same time. Especially, we do not select the keyphrase pairs whose score is below to 0.05 or above to 0.95, because these pairs are either completely unrelated or exactly the same. We randomly divide the data into five samples and ask five different people to rate each pair of keyphrases (Scores range: 0.0, 0.1, ... , 0.9, 1.0). Both of the annotators have no less than a bachelor degree, which have the enough ability of evaluating the quality of model predictions. Then we used **Pearson** correlation coefficient and **Spearman** correlation coefficient to measure the effect of the BERT Scorer. The human evaluation results are shown in Table 3. From the results, we can conclude that the scorer produced by BERT has high quality, and hence, it can act as a helpful signal during our training process.

Annotator	Pearson	Spearman
People 1	0.894	0.884
People 2	0.881	0.867
People 3	0.874	0.856
People 4	0.889	0.873
People 5	0.883	0.875
Total	0.884	0.870

Table 3: The results of manually evaluation on Pearson and Spearson correlation coefficient.

6 Conclusion

In this paper, we utilize a two-stage reinforcement learning training framework with a fine-grained evaluation metric. We propose the FG-score or the continuous BERT-score as the reward in the first-stage training, which improves the generation quality on token-level and then beneficial to the second-stage training. Experiments on KG benchmarks show the effectiveness of our proposed method, and then we also demonstrated the contribution of each module in the FG function. In addition, we evaluate the performance of BERT-based scorer manually. In future work, we will consider improving the training of BERT scorer's performance and making the two-stage RL training more effective.

Acknowledgement

We thank the anonymous reviewers for their helpful comments. We also thank Xiaoyu Xing for her valuable feedback about the paper presentation. This work was partially funded by China National Key R&D Program (No. 2017YFB1002104), National Natural Science Foundation of China (No. 61976056, 62076069), Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103).

References

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

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2163–2174. Association for Computational Linguistics.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4057–4066. 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, ACL 2020, Online, July 5-10, 2020*, pages 1095–1105. Association for Computational Linguistics.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2019. Title-guided encoding for keyphrase generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6268–6275. AAAI Press.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

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 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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Anette Hulth and Beáta Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*, page 537–544. The Association for Computer Linguistics.

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

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

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yichao Luo, Zhengyan Li, Bingning Wang, Xiaoyu Xing, Qi Zhang, and Xuanjing Huang. 2020. SenSeNet: Neural keyphrase generation with document structure. *arXiv preprint arXiv:2012.06754*.

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, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421. The 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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 582–592. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1747–1759. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1179–1195. IEEE Computer Society.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

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, Sofia, Bulgaria. Association for Computational Linguistics.

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

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3612–3621. Association for Computational Linguistics.

Yige Xu, Yichao Luo, Yicheng Zou, Zhengyan Li, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. 2021. Searching effective transformer for seq2seq keyphrase generation. In *Natural Language Processing and Chinese Computing - 10th CCF International Conference, NLPCC 2021*.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4598–4608. 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, ACL 2020, Online, July 5-10, 2020*, pages 7961–7975. Association for Computational Linguistics.

Jing Zhao, Junwei Bao, Yifan Wang, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021. SGG: learning to select, guide, and generate for keyphrase generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5717–5726. Association for Computational Linguistics.

Jing Zhao and Yuxiang Zhang. 2019. Incorporating linguistic constraints into keyphrase generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5224–5233. Association for Computational Linguistics.