WR-ONE2SET: Towards Well-Calibrated Keyphrase Generation

Binbin Xie[1,3], **Xiangpeng Wei**[2], **Baosong Yang**[2], **Huan Lin**[2], **Jun Xie**[2], **Xiaoli Wang**[3], **Min Zhang**[4] and **Jinsong Su**[1,3*]
[1]School of Informatics, Xiamen University, China [2]Alibaba Group, China
[3]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, China
[4]Soochow University, China
xdblb@stu.xmu.edu.cn pemywei@gmail.com
yangbaosong.ybs@alibaba-inc.com sjs@xmu.edu.cn

Abstract

Keyphrase generation aims to automatically generate short phrases summarizing an input document. The recently emerged ONE2SET paradigm (Ye et al., 2021) generates keyphrases as a set and has achieved competitive performance. Nevertheless, we observe serious calibration errors outputted by ONE2SET, especially in the *over-estimation* of ∅ token (means "no corresponding keyphrase"). In this paper, we deeply analyze this limitation and identify two main reasons behind: 1) the parallel generation has to introduce excessive ∅ as padding tokens into training instances; and 2) the training mechanism assigning target to each slot is unstable and further aggravates the ∅ token over-estimation. To make the model well-calibrated, we propose WR-ONE2SET which extends ONE2SET with an adaptive instance-level cost **W**eighting strategy and a target **Re**-assignment mechanism. The former dynamically penalizes the over-estimated slots for different instances thus smoothing the uneven training distribution. The latter refines the original inappropriate assignment and reduces the supervisory signals of over-estimated slots. Experimental results on commonly-used datasets demonstrate the effectiveness and generality of our proposed paradigm.

1 Introduction

Keyphrases are short phrases fully encoding the main information of a given document. They can not only facilitate readers to quickly understand the document, but also provide useful information to many downstream tasks, including document classification (Hulth and Megyesi, 2006), summarization (Wang and Cardie, 2013), etc.

With the rapid development of deep learning, *keyphrase generation* (Meng et al., 2017) has attracted increasing attention due to its ability to produce phrases that even do not match any contiguous subsequence of the source document.[1] Dominant models of keyphrase generation

are constructed under three paradigms: ONE2ONE (Meng et al., 2017), ONE2SEQ (Yuan et al., 2020) and ONE2SET (Ye et al., 2021). Among these paradigms, ONE2SET exhibits the state-of-the-art (SOTA) performance. As illustrated in Figure 1, it considers keyphrase generation as a set generation task. After padding keyphrases to a fixed number with special token ∅, they define multiple slots that individually generate each keyphrase in parallel. During training, each slot is assigned with a keyphrase or ∅ token[2] via a K-step target assignment mechanism. Specifically, the model first generates K tokens from each slot and then determines the optimal target assignment using a bipartite matching algorithm (Kuhn, 2010). The superiority of ONE2SET stems from its conditional independence, that is, the prediction distribution of each slot depends only on the given document other than the order of keyphrases like

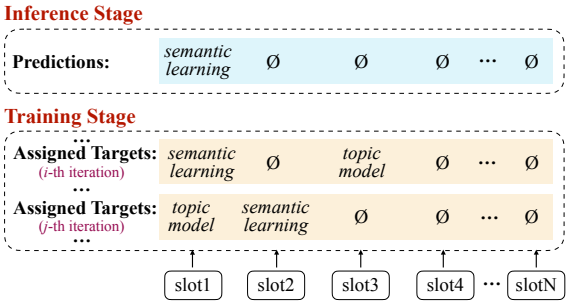


Figure 1: An example of ONE2SET paradigm at training and inference stages. "Assigned Targets (∗-th iteration)" represents the multiple feasible target permutations generated by K-step target assignment mechanism at different training iterations. In this case, both "slot2" and "slot3" are expected to generate keyphrases. However, they often use ∅ token as supervisory signals, and thus over-estimate and output ∅ token.

* Corresponding author.

[1]An example is shown in Appendix, Table A.1.

[2]In this work, we define that the keyphrase can not be a ∅ token.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 7283–7293
December 7-11, 2022

ONE2SEQ. This is more compatible with the unordered property of keyphrases and decreases the difficulty of the model training (Ye et al., 2021).

Despite of its success, we observe serious over-estimation problem on ∅ token, which significantly affects the generation quality. For example, in Figure 1, both "slot2" and "slot3" are expected to generate keyphrases, but ∅ token is over-confidently given. Two questions naturally arise: 1) *what are reasons behind the over-estimation problem in* ONE2SET*?* and 2) *how can we alleviate them?*

In order to answer the first question, we conduct extensive analyses, and conclude two reasons. Firstly, the over-estimation is a by-product inherently carried by the parallel generation. More concretely, excessive ∅ tokens have been introduced as the padding tokens and served as supervisory signals in training data. The unbalanced data and the lack of dependency among slots leads each slot to learn to commonly generate ∅ token. Secondly, the K-step target assignment mechanism provides multiple feasible target permutations that are assigned to slots. As shown in Figure 1, the targets of the given document can be assigned in different permutation at each training iteration, which further increases the probability of ∅ token to be assigned as supervisory signal for each slot, thus exacerbating the over-estimation problem. Both problems make the learned probabilities of the assigned targets deviate from its ground truth likelihood, finally constructing a miscalibrated model.

Consequently, we approach the above problems from the calibration perspective and propose two strategies that extend ONE2SET to WR-ONE2SET. Specifically, an *adaptive instance-level cost weighting* is first introduced to penalize the over-estimated slots of different instances. According to the seriousness of the issue, instances are rendered different weights, therefore dynamically balancing the model training. Besides, we propose a *target re-assignment* mechanism to refine the original inappropriate assignment and reduce the supervisory signals of ∅ token. In particular, we re-assign targets for the slots potentially generating fresh keyphrases but being pre-assigned with ∅ token. In these ways, WR-ONE2SET is encouraged to produce well-calibrated probabilities on keyphrase generation. Overall, major contributions of our work are three-fold:

- Through in-depth analyses, we point out that the advanced keyphrase generation architecture ONE2SET suffers from the ∅ token over-estimation, which is inherently caused by its parallism and the target assignment mechanism.
- We propose WR-ONE2SET which enhances the original framework with two effective strategies to calibrate the over-estimation problem from the training perspective.
- Extensive experiments on five widely-used datasets reveal the universal-effectiveness of our model.
- We release our code at `https://github.com/DeepLearnXMU/WR-One2Set`.

2 Related Work

Early studies mainly focus on automatic keyphrase extraction (Hulth, 2003; Mihalcea and Tarau, 2004; Nguyen and Kan, 2007; Wan and Xiao, 2008), which aims to directly extract keyphrases from the input document. Recently, with the rapid development of deep learning, neural network-based models have been widely used in keyphrase generation. Typically, these models are based on an attentional encoder-decoder framework equipped with copy mechanism, which is able to generate both present and absent keyphrases (Meng et al., 2017). Generally, these models are constructed under the following paradigms: 1) ONE2ONE (Meng et al., 2017; Chen et al., 2019a,b). Under this paradigm, the input document is paired with each target keyphrase to form an independent training instance for model training. During inference, the models are encouraged to produce multiple keyphrases via beam search. 2) ONE2SEQ (Chan et al., 2019; Yuan et al., 2020; Chen et al., 2020; Wu et al., 2021). It considers keyphrase generation as a sequence generation task, where different keyphrases are concatenated into a sequence in a predefined order. In this way, the semantic dependence between keyphrases can be exploited to benefit keyphrase generation. 3) ONE2SET (Ye et al., 2021). Unlike ONE2SEQ, this paradigm considers the keyphrases as a set, which can be predicted from slots in a parallel manner and partial target matching algorithm.

Considering that ONE2ONE neglects the correlation among keyphrases, the most popular paradigm ONE2SEQ exploits the correlation by pre-defining the keyphrases order for model training and inference. Nevertheless, ONE2SEQ is the opposite of the flexible and unordered properties of keyphrases,

increasing the difficulty of the model training. Due to the parallelism and the conditional independence, ONE2SET attracts much attention in the keyphrase generation community, and achieves the SOTA performance. As this method has just been put forward, it is inevitable to exist imperfections. Hence, we are committed to analyses and further optimizing this framework. To the best of our knowledge, this is the first attempt to improve ONE2SET.

3 Background

Here, we briefly introduce SETTRANS (Ye et al., 2021), which is based on the ONE2SET paradigm. It is a Transformer-based, semi-autoregressive model. Typically, it introduces N slots, each of which introduces a learnable control code as the additional decoder input, to generate keyphrases or $\varnothing$ tokens in parallel. Its training involves two stages: 1) a K-step target assignment mechanism is firstly used to determine the correspondence between each prediction and target, and then 2) a new training objective is introduced to optimize the whole model. It contains two set losses to separately deal with two kinds of keyphrases: *present keyphrases* appearing in the input document, and *absent keyphrases* that do not match any contiguous subsequence of the document.

K-Step Target Assignment At this stage, the model predicts K tokens from each slot, where the predicted probability distributions are also collected. Then, an optimal assignment m between predictions and targets can be found by a bipartite matching algorithm (Kuhn, 2010):

$$m = \arg\min_{m \in M(N)} \sum_{i=1}^{N} \mathcal{C}_{\text{match}}(y^{m(i)}, \mathbf{P}^i), \quad (1)$$

where $M(N)$ denotes a set of all N-length target index permutations, and the optimal permutation m can be considered as a mapping function from the slot i to the target index $m(i)$.[3] $\mathcal{C}_{\text{match}}(y^{m(i)}, \mathbf{P}^i)$ is a pair-wise matching loss between the target $y^{m(i)}$ and the predicted probability distributions $\mathbf{P}^i$ of the slot i. Note that the set of targets are also padded to size N with $\varnothing$ tokens.

Model Optimization with Set Losses During the second stage, the model is trained with the sum of two set losses. Concretely, slots are equally split into two sets, dealing with the generations of present and absent keyphrases, respectively. Next, the above target assignment is performed on these two sets separately, forming a mapping m^p for present keyphrases, and a mapping m^a for absent keyphrases. Finally, the training objective becomes

$$\mathcal{L}(\theta) = -\left[\sum_{i=1}^{\frac{N}{2}} \mathcal{L}^p(\theta, y^{m^p(i)}) + \sum_{i=\frac{N}{2}+1}^{N} \mathcal{L}^a(\theta, y^{m^a(i)})\right] \quad (2)$$

$$\mathcal{L}^p(\theta, z) = \begin{cases} \lambda_{pre} \cdot \sum_{t=1}^{|z|} \log \hat{p}_t^i(z_t) & \text{if } z{=}\varnothing \\ \sum_{t=1}^{|z|} \log \hat{p}_t^i(z_t) & \text{otherwise} \end{cases} \quad (3)$$

where λ_{pre} is a hyper-parameter used to reduce the negative effect of excessive $\varnothing$ tokens, z_t symbolizes the t-th token of the target z, and $\hat{p}_t^i$ is the t-th predicted probability distribution of the i-th slot using teacher forcing. Meanwhile, $\mathcal{L}^a(\theta, z)$ is defined in the similar way as $\mathcal{L}^p(\theta, z)$ with a hyper-parameter λ_{abs}.

4 Preliminary Analyses

Although ONE2SET has achieved competitive performance, it still faces one major problem, i.e. *$\varnothing$ token over-estimation. This occurs in such slots that produce $\varnothing$ tokens via the vanilla prediction while are able to generate correct keyphrases through the non-$\varnothing$ prediction*[4]. For illustration, we force all slots to generate non-$\varnothing$ predictions during inference, where 14.6% of slots can produce correct ones. However, if we remove this restriction, 34.5% of these slots directly output $\varnothing$ tokens, revealing the over-estimation of $\varnothing$ token. Such kind of miscalibration (Guo et al., 2017; Kumar and Sarawagi, 2019) is a common drawback in neural network based models, which not only seriously hurts the generation quality of the ONE2SET paradigm, but also limits the users' trust towards it.

To understand the reasons behind this, we use the commonly-used KP20k dataset (Meng et al., 2017) to train a standard SETTRANS model, where the assigned targets to the slots of each instance are recorded during the last 80,000 training steps with an interval of 8,000 steps. Here, we can obtain two crucial observations.

Observation 1: *Excessive $\varnothing$ tokens have been introduced as the padding tokens and served as supervisory signals in training data.* ONE2SET models keyphrase generation in a parallel computation fashion, therefore extensive padding $\varnothing$ to-

[3] Please note that instead of following Ye et al. (2021) to use the function $\pi(i')$ mapping the target index i' to the slot index $\pi(i')$, we use $m(i)$ that is the inverse function of $\pi(i)$, so as to facilitate subsequent descriptions.

[4] When performing the non-$\varnothing$ prediction, we remove $\varnothing$ token from the prediction vocabulary to generate a keyphrase.

Target Type	Pre.KP Slots	Abs.KP Slots
∅	72.4%	80.4%
Target KP	27.6%	19.6%

Table 1: The proportions of ∅ token and target keyphrases used as supervisory signals during training. "KP" means keyphrase. "Pre.KP" and "Abs.KP" represent present and absent keyphrases, respectively.

Instance Type	All KP Slots
Instance(#OV-Slot=0)	42.1%
Instance(#OV-Slot=1)	31.9%
Instance(#OV-Slot=2)	15.5%
Instance(#OV-Slot≥3)	10.5%

Table 2: The proportions of instances involving different numbers of slots over-estimating ∅ token. Instance(#OV-Slot=n) means the instances containing n slots over-estimating ∅ token. Please note that the greater n, the more severe ∅ token over-estimation.

kens are used to make sure the fixed lengths of different samples. Table 1 shows the proportions of ∅ token and target keyphrases involved during the model training. We can observe that on both present and absent keyphrase slots, ∅ token accounts for the vast majority, exceeding 70%. In addition, instances suffer from different degrees of ∅ token over-estimation. Table 2 shows the proportions of training instances grouped by the number of slots over-estimating ∅ token. We can find that the instances (e.g. Instance(#OV-Slot≥1)) account for significant proportions, and exist varying degrees of ∅ token over-estimation.

Observation 2: *The K-step assignment mechanism is unstable and further increases the possibility of ∅ tokens being served as supervisory signals for some slots.* In spite of the clever design of K-step assignment mechanism, it unstably provides different feasible target permutations to slots at the training time. We argue that this further widens the gap between the distribution of supervisory signals and that of the ground-truth.

To illustrate this, we classify the slots of each instance into three categories according to its target assignments: 1) Slot(∅), each slot of this category is always assigned with ∅ tokens. Apparently, these slots hardly generate keyphrases after training; 2) Slot(Target KP), each slot of this category is always assigned with target keyphrases and thus it has high probability of generating a keyphrase; 3) Slot(∅+Target KP), each slot is assigned with

Slot Type	Pre.KP Slots	Abs.KP Slots
Slot(∅)	61.2%	66.4%
Slot(Target KP)	17.6%	9.3%
Slot(∅+Target KP)	21.2%	24.4%

Table 3: The proportions of slots with different target assignments during the model training. Slot(∅+Target KP) means the slots are assigned with ∅ tokens and target keyphrases alternatively at different iterations. Note that the higher proportions of Slot(∅+Target KP), the more slots contain unstable supervisory signals.

target keyphrases or ∅ tokens at different iterations during model training. From Table 3, we can observe that on both present and absent keyphrase slots, the proportions of Slot(∅+Target KP) are quite high, exceeding those of Slot(Target KP). Quite evidently, the supervisory signals of slots in Slot(∅+Target KP) are unstable. Those slots that should be labeled with Target KP are assigned with ∅ token, further decreasing the probabilities of these slots generating keyphrases.

5 WR-ONE2SET

As discussed above, the parallelism and the training mechanism of ONE2SET bring the advantages of conditional independence, but inherently lead to the miscalibration of the model. Our principle is to maintain the primary advantages, and meanwhile, calibrating the model with lightweight strategies. To this end, we propose WR-ONE2SET that significantly extends the conventional ONE2SET paradigm in two training aspects, including an *adaptive instance-level cost weighting strategy*, and a *target re-assignment mechanism*.

To facilitate subsequent descriptions, we summarize all related formal definitions in Appendix, Table A.2 for better understanding this paradigm.

5.1 Adaptive Instance-Level Cost Weighting

Connection to Observation 1. As analyzed previously, excessive ∅ tokens lead to the over-estimation of ∅ token. Although SETTRANS introduces hyper-parameters λ_{pre} and λ_{abs} to adjust the training loss of conventional ONE2SET paradigm, such fixed hyper-parameters are still unable to deal with this issue well due to the different degrees of ∅ token over-estimation in different training instances.

We alternatively develop an adaptive instance-level cost weighting strategy to dynamically scale the losses corresponding to ∅ tokens, alleviating

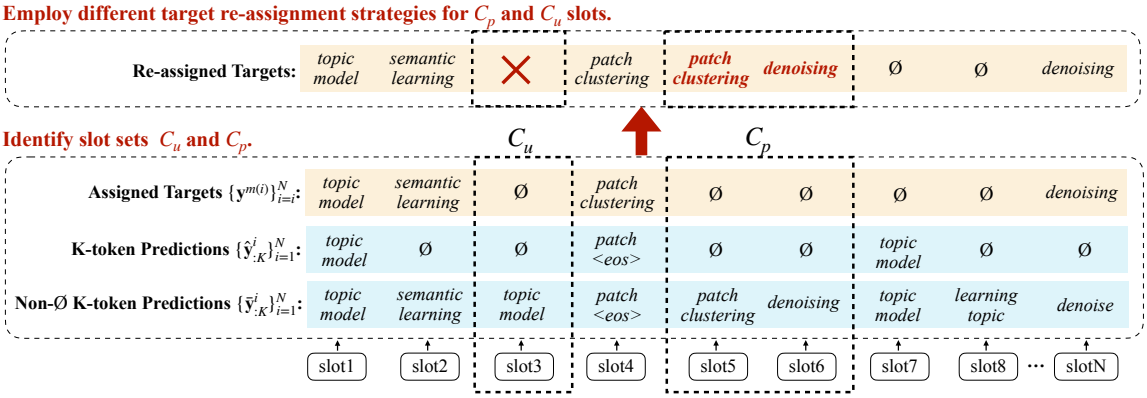


Figure 2: The procedure of target re-assignment involving two steps: identifying the potential slot set C_p and the unimportant one C_u, and then employing different re-assignment operations to deal with them, respectively. Here, "×" represents assigning no supervisory signal to the slots of C_u, and following Ye et al. (2021), we set $K = 2$.

the class imbalance of training data. Concretely, we first identify a set of slots, denoted as $C_{!\varnothing}$, where each slot is assigned with a keyphrase as supervisory signal. Intuitively, for each slot i in $C_{!\varnothing}$, the degree of $\varnothing$ token over-estimation is related to its two predicted probabilities using teacher forcing (See Section 4): 1) $\hat{p}^i(y_0^{m(i)})$, symbolizing the predicted probability of the first token of assigned target, and 2) $\hat{p}^i(\varnothing)$, denoting the predicted probability of $\varnothing$ token. Thus, we directly use the ratio between $\hat{p}^i(y_0^{m(i)})$ and $\hat{p}^i(\varnothing)$ to approximately quantify the degree of $\varnothing$ token over-estimation for training efficiency. Furthermore, we define this degree for each instance as

$$\lambda_{adp} = \frac{1}{|C_{!\varnothing}|} \cdot \sum_{i \in C_{!\varnothing}} \min\left(\frac{\hat{p}^i(y_0^{m(i)})}{\hat{p}^i(\varnothing)}, 1\right). \quad (4)$$

Note that for each slot i in $C_{!\varnothing}$, if its predicted probability $\hat{p}^i(y_0^{m(i)})$ is greater than $\hat{p}^i(\varnothing)$, it is considered to have no $\varnothing$ token over-estimation, and we directly limit its ratio to 1.

Finally, we adjust the hyper-parameters λ_{pre} and λ_{abs} of Equation 3 into $\lambda_{adp} \cdot \lambda_{pre}$ and $\lambda_{adp} \cdot \lambda_{abs}$ for each training instance, respectively. Note that, λ_{adp} is dynamically updated during the training process, and thus is more general for model training compared with fixed hyper-parameters.

5.2 Target Re-Assignment

Connection to Observation 2. Due to the effect of K-step target assignment mechanism, many slots are alternatively assigned with target keyphrases and $\varnothing$ tokens, which decreases the probabilities of these slots generating correct keyphrases.

We propose a target re-assignment mechanism to alleviate this issue. As shown in the upper part of Figure 2, during the process of K-step target assignment, we first record three kinds of phrases for each slot i: 1) $y^{m(i)}$, the assigned target of the slot i; 2) $\hat{y}_{:K}^i$, the first K tokens of the vanilla prediction from the slot i. Note that $\hat{y}_{:K}^i$ may be a $\varnothing$ token; and 3) $\bar{y}_{:K}^i$, the first K tokens of the non-$\varnothing$ prediction from the slot i.

Here, we mainly focus on the slots, each of which is assigned with ϕ token as supervisory signals and its non-ϕ K-token prediction is consistent with some targets. For such slot i, if its non-$\varnothing$ K-token prediction $\bar{y}_{:K}^i$ is totally different from all K-token predictions $\{\hat{y}_{:K}^i\}_{i=1}^N$, we consider it has the potential to generate a fresh keyphrase and boost the model performance. Thus, we include it into the **potential slot set** C_p. By contrast, if its $\bar{y}_{:K}^i$ has occurred in the set of $\{\hat{y}_{:K}^i\}_{i=1}^N$, we regard it as an unimportant slot without effect on the model performance, and add it into the **unimportant slot set** C_u. Back to Figure 2, we observe that the non-$\varnothing$ K-token prediction of "slot3" is "*topic model*", which is also the K-token prediction of "slot1" and "slot7". Thus, "slot3" is an unimportant slot. Meanwhile, both "slot5" and "slot6" are potential slots.

Then, as illustrated in the lower part of Figure 2, we employ two target re-assignment operations to deal with the above two kinds of slots, respectively: 1) we re-assign each slot of C_p with its best-matched target keyphrase, so as to increase the probability of this slot generating the target keyphrase; and 2) we assign no target to each slot of C_u, which alleviates the problem that the same target is assigned to different slots as supervisory signals. In this way, the training losses of slots in C_u will be masked at this training iteration. Let us

Model	Inspec		NUS		Krapivin		SemEval		KP20k	
	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$
Existing Neural Keyphrase Generation Models										
CATSEQ(R)	0.225	0.262	0.323	0.397	0.269	0.354	0.242	0.283	0.291	0.367
CATSEQ	0.281_{5}	0.325_{6}	0.370_{7}	0.419_{10}	0.315_{8}	0.365_{5}	0.287_{14}	0.325_{15}	0.332_{1}	0.377_{1}
UNIKEYPHRASE	0.260	0.288	0.415	0.443	—	—	0.302	0.322	0.347	0.352
SETTRANS	0.285_{3}	0.324_{3}	0.406_{12}	0.450_{7}	0.326_{12}	0.364_{12}	0.331_{20}	0.357_{13}	0.358_{5}	0.392_{4}
PROMPTKP	0.260	0.294	0.412	0.439	—	—	0.329	0.356	0.351	0.355
Our Models										
SETTRANS	0.282_{2}	0.320_{2}	0.399_{5}	0.437_{8}	0.334_{7}	0.368_{4}	0.333_{6}	0.357_{4}	0.359_{2}	0.392_{2}
SETTRANS(w/o λ_{pre}, λ_{abs})	0.100_{4}	0.148_{6}	0.173_{18}	0.258_{42}	0.155_{9}	0.280_{16}	0.129_{7}	0.188_{8}	0.191_{7}	0.321_{9}
SETTRANS(#SLOT=12)	0.280_{1}	0.316_{2}	0.387_{9}	0.423_{4}	0.324_{5}	0.369_{3}	0.302_{6}	0.327_{4}	0.347_{3}	0.385_{3}
SETTRANS(#SLOT=16)	0.280_{7}	0.318_{6}	0.384_{7}	0.431_{10}	0.319_{3}	0.362_{1}	0.316_{15}	0.356_{17}	0.342_{2}	0.382_{2}
SETTRANS(#SLOT=24)	0.284_{13}	0.320_{18}	0.400_{1}	0.453_{1}	0.331_{1}	0.367_{5}	0.327_{18}	0.359_{11}	0.360_{1}	$\mathbf{0.395}_{1}$
SETTRANS(#SLOT=28)	0.277_{9}	0.317_{8}	0.402_{1}	$\mathbf{0.454}_{4}$	0.338_{4}	$\mathbf{0.374}_{6}$	0.315_{3}	0.349_{2}	0.355_{4}	0.392_{4}
SETTRANS(w/ BATCHING)	0.281_{7}	0.271_{6}	0.379_{4}	0.358_{4}	0.316_{2}	0.264_{7}	0.300_{8}	0.293_{6}	0.341_{2}	0.304_{3}
OUR MODEL	$\mathbf{0.330}_{3}$‡	$\mathbf{0.351}_{3}$‡	$\mathbf{0.428}_{5}$‡	0.452_{1}	$\mathbf{0.360}_{4}$‡	0.362_{5}	$\mathbf{0.360}_{5}$‡	$\mathbf{0.370}_{2}$‡	$\mathbf{0.370}_{1}$	0.378_{2}

Table 4: Results of present keyphrase prediction. Results shown in the upper part are directly cited from their corresponding papers. The subscript denotes the corresponding standard deviation (e.g., 0.330_{3} indicates 0.330±0.003). ‡ indicates significant at p<0.01 over SETTRANS with 1,000 booststrap tests (Efron and Tibshirani, 1993).

revisit Figure 2, we re-assign "slot5" with "*patch clustering*", "slot6" with "*denoising*" and no supervisory signal to "slot3". Through the above process, we can convert the original target assignment m into a new one, where we use the conventional training objective (See Equation 2) adjusted with λ_{adp} (See Equation 3) to train our model.

6 Experiments

6.1 Setup

Datasets. We train various models and select the optimal parameters on the KP20k validation dataset (Meng et al., 2017). Then, we evaluate these models on five test datasets: Inspec (Hulth, 2003), NUS (Nguyen and Kan, 2007), Krapivin (Krapivin et al., 2009), SemEval (Kim et al., 2010), and KP20k. As implemented in (Yuan et al., 2020; Ye et al., 2021), we perform data preprocessing including tokenization, lowercasing, replacing all digits with the symbol $\langle digit \rangle$ and removing duplicated instances.

Baselines. We compare our WR-ONE2SET based model with the following baselines:

- **CATSEQ(R)** (Yuan et al., 2020). This is the most popular RNN-based model trained under the ONE2SEQ paradigm, formulating keyphrase generation as a sequence-to-sequence generation task.
- **CATSEQ** (Ye et al., 2021). It is also trained under the ONE2SEQ paradigm, but utilizing Transformer as backbone.
- **UNIKEYPHRASE** (Wu et al., 2021). This is a large-scale pre-trained language model trained to extract and generate keyphrases jointly.
- **SETTRANS** (Ye et al., 2021). It is our most important baselines. Besides, we report the performance of three SETTRANS variants: **SETTRANS(w/o λ_{pre}, λ_{abs})** that does not introduce any hyper-parameter to alleviate the negative effect of excessive $\varnothing$ tokens, **SETTRANS(#SLOT=N)** that is equipped with $N/2$ and $N/2$ slots for present target keyphrases and absent ones, respectively, and **SETTRANS(w/ BATCHING)** which sorts all training instances in the increasing order of target keyphrase numbers and uses batch-wise randomized order to keep the padding length optimized.
- **PROMPTKP** (Wu et al., 2022). It firstly extracts keywords for automatic prompt construction, and then uses a mask-predict-based approach to generate the final absent keyphrase constrained by prompt.

Implementation Details. We use Transformer-*base* (Vaswani et al., 2017) to construct all models. During training, we choose the top 50,002 frequent tokens to form the predefined vocabulary. We use the Adam optimizer with a learning rate of 0.0001, and a batch size of 12. During inference, we employ greedy search to generate keyphrases.

Model	Inspec		NUS		Krapivin		SemEval		KP20k	
	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$
Existing Neural Keyphrase Generation Models										
CATSEQ(R)	0.004	0.008	0.016	0.028	0.018	0.036	0.016	0.028	0.015	0.032
CATSEQ	0.010_2	0.019_4	0.028_2	0.048_2	0.032_1	0.060_4	0.020_5	0.023_3	0.023_1	0.046_1
UNIKEYPHRASE	0.012	0.022	0.026	0.037	—	—	0.022	0.029	0.032	0.058
SETTRANS	0.021_1	$\mathbf{0.034}_3$	0.042_2	0.060_4	0.047_7	0.073_{11}	0.026_3	0.034_5	0.036_2	0.058_3
PROMPTKP	0.017	0.022	0.036	0.042	—	—	0.028	0.032	0.032	0.042
Our Models										
SETTRANS	0.020_3	0.031_4	0.044_5	0.061_8	0.050_2	0.073_1	0.030_2	0.037_1	0.038_1	0.059_1
SETTRANS(w/o λ_{pre}, λ_{abs})	0.000_0	0.000_0	0.002_1	0.003_2	0.004_1	0.008_2	0.002_1	0.003_2	0.002_1	0.005_1
SETTRANS(#SLOT=12)	0.016_3	0.027_6	0.042_7	0.065_{13}	0.047_5	0.073_9	0.024_8	0.031_8	0.033_1	0.057_2
SETTRANS(#SLOT=16)	0.018_1	0.030_2	0.040_5	0.060_8	0.045_3	$\mathbf{0.074}_2$	0.023_1	0.031_1	0.034_1	0.057_2
SETTRANS(#SLOT=24)	0.019_2	0.029_5	0.044_3	0.061_4	0.046_5	0.073_9	0.026_3	0.035_4	0.038_1	0.059_2
SETTRANS(#SLOT=28)	0.016_3	0.026_5	0.044_4	0.063_3	0.043_1	0.070_1	0.021_4	0.027_2	0.032_3	0.054_3
SETTRANS(w/ BATCHING)	0.023_1	0.030_4	0.050_5	0.067_6	0.049_3	0.059_9	0.034_4	0.038_6	0.045_2	0.058_2
OUR MODEL	$\mathbf{0.025}_2$	$\mathbf{0.034}_4$	$\mathbf{0.057}_5\ddagger$	$\mathbf{0.071}_3\ddagger$	$\mathbf{0.057}_1\ddagger$	$\mathbf{0.074}_2$	$\mathbf{0.040}_3\ddagger$	$\mathbf{0.043}_5\ddagger$	$\mathbf{0.050}_1\ddagger$	$\mathbf{0.064}_2$

Table 5: Results of absent keyphrase prediction.

To ensure a fair comparison with SETTRANS, we also set both slot numbers for present and absent keyphrases as 10, the target assignment step K as 2, λ_{pre} as 0.2 and λ_{abs} as 0.1, respectively. Particularly, we run all experiments three times with different random seeds and report the average results, so as to alleviate the impact of the instability of model training.

Evaluation Metrics. Following previous studies (Chen et al., 2020; Ye et al., 2021), we use macro averaged $F_1@5$ and $F_1@M$ to evaluate the quality of both present and absent keyphrases. When using $F_1@5$, if the prediction number is less than five, blank keyphrases are added to make the keyphrase number reach five. Particularly, we employ the Porter Stemmer[5] to remove the identical stemmed keyphrases.

6.2 Main Results

Table 4 and Table 5 show the prediction results on present and absent keyphrases, respectively. We can draw the following conclusions:

First, our reproduced SETTRANS achieves comparable performance to Ye et al. (2021). Second, when removing both λ_{pre} and λ_{abs} from SETTRANS, its performance significantly drops, showing that the $\varnothing$ token over-estimation severely limits its full potential. Third, we observe no improvements with different number of slots. Fourth, the commonly-used batching method for sequence generation is not beneficial for SETTRANS. Finally, our model significantly surpasses all baselines. These results strongly validate the effectiveness and generalization of our WR-ONE2SET paradigm.

[5] https://github.com/nltk/nltk/blob/develop/nltk/stem/porter.py

Model	In-domain		Out-domain	
	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$
Present Keyphrase Prediction				
OUR MODEL	**0.370**	0.378	**0.370**	**0.384**
w/o RE-ASSIGN	0.368	0.375	0.360	0.377
w/o WEIGHTING	0.365	**0.393**	0.340	0.374
RE-ASSIGN⇒RAND-ASSIGN	0.368	0.377	0.365	0.380
SETTRANS	0.359	0.392	0.336	0.373
Absent Keyphrase Prediction				
OUR MODEL	**0.050**	**0.064**	**0.043**	**0.055**
w/o RE-ASSIGN	0.047	0.062	0.042	0.053
w/o WEIGHTING	0.043	0.063	0.039	0.052
RE-ASSIGN⇒RAND-ASSIGN	0.048	0.063	0.042	0.053
SETTRANS	0.038	0.059	0.034	0.052

Table 6: Ablation study on keyphrase predictions.

6.3 Ablation Study

To better investigate the effectiveness of our proposed strategies on WR-ONE2SET, we report the performance of variants of our model on two test sets: 1) KP20k that is an **in-domain** one, and 2) the combination of Inspec, NUS, Krapivin and SemEval, which is **out-domain**. Here, we mainly consider three variants: 1) w/o RE-ASSIGN, which removes the target re-assignment mechanism from our model; and 2) w/o WEIGHTING. It discards the adaptive instance-level cost weighting strategy;

Model	In-domain	Out-domain
SETTRANS(w/o λ_{pre}, λ_{abs})	0.747	0.809
SETTRANS	0.345	0.418
SETTRANS(w/ RE-ASSIGN)	0.301	0.386
SETTRANS(w/ WEIGHTING)	0.240	0.308
OUR MODEL	**0.211**	**0.263**

Table 7: The proportions of slots over-estimating ∅ token.

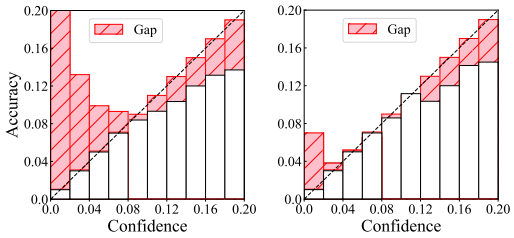
and 3) RE-ASSIGN⇒RAND-ASSIGN. This variant randomly re-assigns targets to the slots in C_p.

As shown in Table 6, when removing the target re-assignment mechanism, we observe a performance degradation on keyphrase predictions. Likewise, the variant w/o WEIGHTING is obviously inferior to our model on most metrics. Therefore, we believe that our proposed strategies indeed benefit the generation of keyphrase set.

6.4 Analyses of ∅ Token Over-Estimation

We also compare various models according to the proportion of slots over-estimating ∅ tokens. Here, the proportion is the ratio between two slot numbers obtained from the whole training data: one is the number of slots that directly output ∅ token via the vanilla prediction while generating correct keyphrases through the non-∅ prediction; and the other is the number of slots that generate correct keyphrases via the non-∅ prediction. Table 7 displays the results. The proportions of SETTRANS (w/o λ_{pre}, λ_{abs}) exceeds 70%, demonstrating the severe ∅ token over-estimation of the ONE2SET paradigm. By comparison, the proportions of SETTRANS decrease, validating the effectiveness of fixed hyper-parameters λ_{pre} and λ_{abs} on alleviating the class imbalance of training data. Moreover, whether adaptive instance-level cost weighting strategy or target re-assignment mechanism is used alone, the proportions of SETTRANS can be further reduced. Particularly, our model achieves the lowest proportions, proving that our strategies can complement each other.

Besides, following Guo et al. (2017), we show the reliability diagram of SETTRANS and our model in Figure 3. It displays the relationship between the prediction confidence (the predicted probability of model) and the prediction accuracy within the confidence interval [0, 0.2]. Especially, the predictions within the confidence interval [0, 0.2] account for 69.8% of all predictions. Please note that, if a model is well-calibrated, the gap between the confidence and the accuracy will be small. Overall, the gap of our model is less than that of SETTRANS, which demonstrates that our proposed strategies can calibrate the predictions with low confidence.



(a) SETTRANS (b) OUR MODEL

Figure 3: Reliability diagrams of SETTRANS and our model on the in-domain test set. "Gap" (areas marked with slash) denotes the difference between the prediction confidence and the prediction accuracy. Smaller gaps denote better calibrated outputs.

Model	In-domain			Out-domain		
	#Pre	#Abs	Dup	#Pre	#Abs	Dup
ORACLE	3.31	1.95	-	5.53	3.51	-
CATSEQ(R)	**3.71**	0.55	0.39	3.46	0.72	0.54
CATSEQ	4.64	1.16	0.26	4.34	1.28	0.38
SETTRANS	5.10	**2.01**	**0.08**	4.62	2.18	**0.08**
SETTRANS(w/ RE-ASSIGN)	5.40	2.64	0.10	4.83	2.72	0.09
SETTRANS(w/ WEIGHTING)	6.19	3.12	0.11	**5.70**	**3.56**	0.09
OURS MODEL	6.35	3.26	0.10	5.94	3.60	0.10

Table 8: Numbers and duplication ratios of predicted keyphrases on test datasets. "ORACLE" refers to the average number of target keyphrases.

6.5 Diversity of Predicted Keyphrases

Follow previous studies (Chen et al., 2020; Ye et al., 2021), we report the average numbers of unique present and absent keyphrases, and the average duplication ratios of all predicted keyphrases, so as to investigate the ability of our model in generating diverse keyphrases,

Table 8 reports the results. As expected, our model generates more keyphrases than previous models and achieves a slightly higher duplication ratio than SETTRANS, however, significantly lower than ONE2SEQ-based models. Note that compared to SETTRANS, the $F_1@5$ and $F_1@M$ scores of our model are significantly improved, which demonstrates that our model performs much better on keyphrase generation.

Slot Type	SETTRANS	OUR MODEL
Present Keyphrase Prediction		
Slot(∅)	61.2%	63.5% (+2.3%)
Slot(Target KP)	17.6%	21.8% (+4.2%)
Slot(∅+Target KP)	21.2%	14.7% (-6.5%)
Absent Keyphrase Prediction		
Slot(∅)	66.4%	69.6% (+3.2%)
Slot(Target KP)	9.3%	12.6% (+3.3%)
Slot(∅+Target KP)	24.4%	17.8% (-6.6%)

Table 9: The proportions of slots with different target assignments for keyphrase predictions.

6.6 Analyses of Target Re-Assignment

Here, we still focus on the assigned targets during the model training mentioned at the beginning of Section 4 and conduct two types of analyses to better understand the effects of our mechanism.

First, we count the proportions of ∅ tokens in assigned targets. Specially, the assigned ∅ tokens accounts for 72.4% and 80.4% on present and absent keyphrase slots, respectively, but decrease to 67.6% and 72.3% in our model. Second, as implemented in Section 4, we still classify the instance slots into three categories and report their proportions in Table 9. We can find the proportions of Slots(∅+KP), where slots are assigned with target keyphrase and ∅ token at different iterations of model training, sharply decline. Besides, for each slot, we use the entropy of target assignment distribution to measure the stability of its supervisory signals. Furthermore, we average the entropy values of all slots to quantify the stability of supervisory signals for each instance. Consequently, we find that the entropy decreases in 68.2% of instances, increases in 26.3% of instances, and remain unchanged in 5.5% of instances. These results indicate that our target re-assignment mechanism indeed not only reduces excessive target ∅ tokens, but also alleviates the instability of supervisory signals.

7 Conclusion

In this paper, we in-depth analyze the serious calibration errors of the ONE2SET paradigm and point out its underlying reasons. To deal with this issue, we then significantly extend the conventional ONE2SET into the WR-ONE2SET paradigm with an adaptive instance-level cost weighting strategy and a target re-assignment mechanism. Extensive experiments verify the effectiveness and generality of our extended paradigm.

In the future, we plan to further refine our WR-ONE2SET paradigm by considering the semantic relation between keyphrases. Besides, we will improve our model by introducing variational neural networks, which have been successfully applied in many NLP tasks (Zhang et al., 2016a,b; Su et al., 2018a,b; Liang et al., 2022). Finally, we will leverage the abundant knowledge from pre-trained models to further enhance our model.

Limitations

As mentioned above, serious ∅ token overestimation problem exists in ONE2SET paradigm, leading to a miscalibrated model. To solve this problem, we propose several strategies based on conventional ONE2SET using the same fixed hyper-parameters as Ye et al. (2021). However, hyper-parameter selection is a labor-intensive, manual, time-consuming process and affects generation performance deeply. Thus, our future work will focus on exploring a parameter-free method. Besides, despite achieving impressive performance, our WR-ONE2SET paradigm is only conducted based on the Transformer, so that it is essential to leverage the abundant knowledge from pre-trained models for better document modeling and keyphrase generation.

Acknowledgements

The project was supported by National Natural Science Foundation of China (No. 62276219, No. 62036004), Natural Science Foundation of Fujian Province of China (No. 2020J06001), and Youth Innovation Fund of Xiamen (No. 3502Z20206059). We also thank the reviewers for their insightful comments.

References

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proc. of ACL*, pages 2163–2174.

Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. 2019a. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. In *Proc. of NAACL-HLT*, pages 2846–2856.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proc. of ACL*, pages 1095–1105.

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

Bradley Efron and Robert Tibshirani. 1993. *An Introduction to the Bootstrap*. Springer.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proc. of ICML*, pages 1321–1330.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proc. of EMNLP*, pages 216–223.

Anette Hulth and Beáta Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proc. of ACL*, pages 537–544.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proc. of SemEval@ACL*, pages 21–26.

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

Harold W. Kuhn. 2010. The hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 29–47.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. In *Proc. of ICLR Debugging Machine Learning Models Workshop*.

Yunlong Liang, Fandong Meng, Chulun Zhou, Jinan Xu, Yufeng Chen, Jinsong Su, and Jie Zhou. 2022. A variational hierarchical model for neural cross-lingual summarization. In *Proc. of ACL,*, pages 2088–2099.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proc. of ACL*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proc. of EMNLP*, pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proc. of ICADL*, pages 317–326.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018a. Variational recurrent neural machine translation. In *Proc. of AAAI*, pages 5488–5495.

Jinsong Su, Shan Wu, Biao Zhang, Changxing Wu, Yue Qin, and Deyi Xiong. 2018b. A neural generative autoencoder for bilingual word embeddings. *Inf. Sci.*, 424:287–300.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS*, pages 5998–6008.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proc. of AAAI*, pages 855–860.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proc. of ACL*, pages 1395–1405.

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 *Proc. of ACL Findings*, pages 825–835.

Huanqin Wu, Baijiaxin Ma, Wei Liu, Tao Chen, and Dan Nie. 2022. Fast and constrained absent keyphrase generation by prompt-based learning. In *Proc. of AAAI*, pages 11495–11503.

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2Set: Generating diverse keyphrases as a set. In *Proc. of ACL*, pages 4598–4608.

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 *Proc. of ACL*, pages 7961–7975.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016a. Variational neural machine translation. In *Proc. of EMNLP*, pages 521–530.

Biao Zhang, Deyi Xiong, Jinsong Su, Qun Liu, Rongrong Ji, Hong Duan, and Min Zhang. 2016b. Variational neural discourse relation recognizer. In *Proc. of EMNLP*, pages 382–391.

A Appendix

A.1 Example

Input Document: an image topic model for image denoising. topic model is a powerful tool for the basic document or image processing tasks. in this study, we introduce a novel image topic model, called latent patch model (lpm), which is a generative bayesian model and assumes that the image and pixels are connected by a latent patch layer. based on the lpm, we further propose an image denoising algorithm namely multiple estimate lpm (melpm). unlike other works, the proposed denoising framework is totally implemented on the latent patch layer, and it is effective for both gaussian white noises and impulse noises. experimental results demonstrate that lpm performs well in representing images...
Keyphrases: topic model; denoising; patch clustering; semantic learning

Table 10: An example of keyphrase generation. The underlined phrases are *present keyphrases* that appear in the document, and other phrases are *absent keyphrases* that do not match any contiguous subsequence of the document.

A.2 Formal Definitions

Symbol	Definition
$\varnothing$	A special token representing no corresponding keyphrase.
N	The predefined number of slots generating keyphrases or $\varnothing$ tokens in parallel.
K	The predefined number of tokens generated from each slot for the conventional target assignment mechanism.
$\mathbf{P}^i$	The predicted probability distributions of the slot i.
$\hat{p}^i(*)$	The predicted probability of a keyphrase from the slot i using teacher forcing. Specially, the j-th token predictive probability of $\hat{p}^i(*)$ is denoted as $\hat{p}^i_j(*)$.
$M(N)$	The set of all N-length target index permutations.
m	The optimal permutation of $M(N)$. It can be considered as a mapping function from the slot i to the target index $m(i)$. Particularly, we use m^p and m^a to denote the optimal permutations for present and absent keyphrases, respectively.
$y^{m(i)}$	The assigned target of the slot i.
λ_{pre}, λ_{abs}	Two predefined hyper-parameters used to reduce the negative effect of excessive $\varnothing$ tokens for present and absent keyphrase predictions, respectively.
λ_{adp}	The degree of $\varnothing$ token over-estimation for each instance, which is leveraged to dynamically scale the losses corresponding to $\varnothing$ tokens in our paradigm.
$\mathcal{L}(\theta)$	The training loss of the whole model with parameters θ. Moreover, we use $\mathcal{L}^p(\theta, z)$ to denote the present keyphrase training loss on the assigned target z. The absent keyphrase training loss $\mathcal{L}^a(\theta, z)$ is defined in a similar way.
$\hat{y}^i_{:K}$	The first K tokens of the prediction from the slot i via the vanilla prediction.
$\bar{y}^i_{:K}$	The first K tokens of the prediction from the slot i through the non-$\varnothing$ prediction, where $\varnothing$ token is removed from the prediction vocabulary.
$C_{!\varnothing}$	The set of slots, each of which is assigned with a keyphrase as supervisory signal.
C_p	The set of potential slots, where each slot has the potential to generate a fresh keyphrase, boosting the performance of the model.
C_u	The unimportant slot set, where each slot has no effect on the model performance.

Table 11: Formal Definitions