Unsupervised KB-to-Text Generation with Auxiliary Triple Extraction using Dual Learning*

Zihao Fu[1], **Bei Shi**[2], **Lidong Bing**[3], **Wai Lam**[1]
[1]Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong
[2]Tencent AI Lab [3]DAMO Academy, Alibaba Group
zhfu@se.cuhk.edu.hk; beishi@tencent.com;
l.bing@alibaba-inc.com; wlam@se.cuhk.edu.hk

Abstract

The KB-to-text task aims at generating texts based on the given KB triples. Traditional methods usually map KB triples to sentences via a supervised seq-to-seq model. However, existing annotated datasets are very limited and human labeling is very expensive. In this paper, we propose a method which trains the generation model in a completely unsupervised way with unaligned raw text data and KB triples. Our method exploits a novel dual training framework which leverages the inverse relationship between the KB-to-text generation task and an auxiliary triple extraction task. In our architecture, we reconstruct KB triples or texts via a closed-loop framework via linking a generator and an extractor. Therefore the loss function that accounts for the reconstruction error of KB triples and texts can be used to train the generator and extractor. To resolve the cold start problem in training, we propose a method using a pseudo data generator which generates pseudo texts and KB triples for learning an initial model. To resolve the multiple-triple problem, we design an allocated reinforcement learning component to optimize the reconstruction loss. The experimental results demonstrate that our model can outperform other unsupervised generation methods and close to the bound of supervised methods.

1 Introduction

Knowledge Base (KB)-to-text task focuses on generating plain text descriptions from given knowledge bases (KB) triples which makes them accessible to users. For instance, given a KB triple <101 Helena, discoverer, James Craig Watson>, it is expected to generate a description sentence such as "101 Helena is discovered by James Craig Watson.". Recently, many research works have been proposed for this task. For example, Gardent et al. (2017a,b) create the WebNLG dataset to generate description for triples sampled from DBPedia (Auer et al., 2007). Lebret et al.'s (2016) method generates people's biographies from extracted Wikipedia infobox. Novikova et al. (2017) propose to generate restaurant reviews by some given attributes and Fu et al. (2020a) create the WikiEvent dataset to generate text based on an event chain. However, the works mentioned above usually map structured triples to text via a supervised seq-to-seq (Sutskever et al., 2014) model, in which large amounts of annotated data is necessary and the annotation is very expensive and time-consuming.

We aim to tackle the problem of completely unsupervised KB-to-text generation which only requires a text corpus and a KB corpus and does not assume any alignment between them. We propose a dual learning framework based on the inverse relationship between the KB-to-text generation task and the triple extraction task. Specifically, the KB-to-text task generates sentences from structured triples while the task of triple extraction extracts multiple triples from plain texts. Such a relationship enables the design of a closed-loop learning framework in which we link KB-to-text generation and its dual task of triple extraction so as to reconstruct the unaligned KB triples and texts. The non-differentiability issue of picking words from our neural model before reconstruction makes it hard to train the extractor or generator effectively using backpropagation. To solve this issue, we apply Reinforcement Learning (RL) based on policy gradients into our dual learning framework to optimize our extractor or generator according to the rewards.

Some semi-supervised works (He et al., 2016; Cao et al., 2019) have been proposed to generate

* The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Codes: 14204418).

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 258–268
December 4 - 7, 2020.

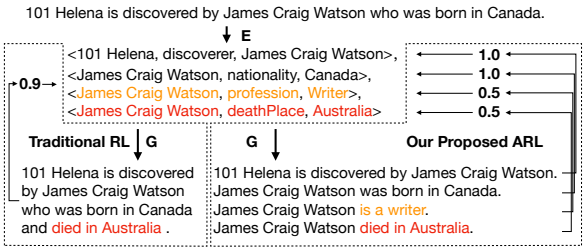

Figure 1: Illustration of the multiple-triple problem, in which ***E*** and ***G*** are extractor and generator respectively. The left part is the traditional RL methods and the right is our proposed ARL method. Four triples are extracted by the extractor. The top two triples are right and the others are wrong. Traditional RL methods give a single reward (0.9) for all the four triples while our proposed ARL gives each triple a different reward. Then the right triples and the wrong triples will be distinguished and optimized differently.

plain texts from data of certain forms in other domains (e.g., translation, semantic parsing) with limited annotated resources. These models contain two major steps. Firstly, they pre-train a weak model based on the labeled data. Secondly, they use an iterative model whose aim is to improve the weak model using the unlabeled data. In each iteration, the input sequence of the original data form is transformed into another form by the original model. Then, it is transformed back to the original data form by an inverse model. However, there are still some challenges applying the existing methods into KB-to-text directly: (1) **Cold start problem.** Existing approaches pre-train the model with labeled data and then fine-tune their models via unlabelled data. Such a mechanism still needs annotated data which is more difficult and expensive to obtain in KB-to-text task. (2) **Multiple-triple problem.** As shown in Fig. 1, multiple triples might be extracted from a text example, and inevitably, the neural extractor could extract some wrong triples. The traditional dual learning approaches (He et al., 2016; Cao et al., 2019), if directly applied, will regard all these triples as one unit and calculate a single reward for all the triples regardless of whether they are correct or not. It not only results in the slow convergence of RL, but also leads to unsatisfactory model performance.

We propose a novel **E**xtractor-**G**enerator **D**ual (EGD) framework which exploits the inverse relationship between KB-to-text generation and auxiliary triple extraction. Our model can resolve the KB-to-text task in a totally unsupervised way. To cope with the cold start problem, we propose a pseudo data generator (PDG) which can generate pseudo text and pseudo KB triples based on the given unaligned KB triples and text respectively with prior knowledge. The extractor and the generator are then pre-trained with the generated pseudo data. To resolve the multiple-triple problem, we propose a novel **A**llocated **R**einforcement **L**earning (ARL) component. Different from traditional RL methods in which one reward is calculated for the whole sequence, ARL allocates different rewards to different sub-parts of the sequence (Fig. 1 right). Therefore, our model can distinguish the quality of each triple and optimize the extractor and the generator more accurately. We compare our framework with existing dual learning methods and the experimental results demonstrate that our model can outperform other unsupervised generation methods and close to the bound of supervised methods.

2 Related Works

Recently many tasks and methods have been proposed to transform existing data into human-readable text. WebNLG (Gardent et al., 2017a,b) is proposed to describe a list of triples sampled from DBPedia (Auer et al., 2007). Except for the KB triples, many other types of data have also been investigated for how to generate text from them. For example, E2E (Novikova et al., 2017) aims at generating text from some restaurants' attributes. Wikibio (Lebret et al., 2016) proposes to generate biographies for the Wikipedia infobox while WikiEvent (Fu et al., 2020a) proposes to generate text based on an event chain. Besides, Chen and Mooney (2008); Wiseman et al. (2017) propose to generate a summarization of a match based on the scores and Liang et al. (2009) propose to generate weather reports based on the records. All these tasks require an elaborately annotated dataset which is very expensive to prepare.

Many methods have been proposed to tackle the dataset insufficiency problem in other tasks. Fu et al. (2020c) propose to directly train the model on partially-aligned data in which the data and the text are not necessarily exactly math, and it can be built automatically. He et al. (2016); Sennrich et al. (2016); Yi et al. (2017) propose dual learning frameworks. They pre-train a weak model with parallel data and refine the model with monolingual data. This strategy has been applied in many related tasks including semantic parsing (Cao et al., 2019), summarization (Baziotis et al., 2019) and

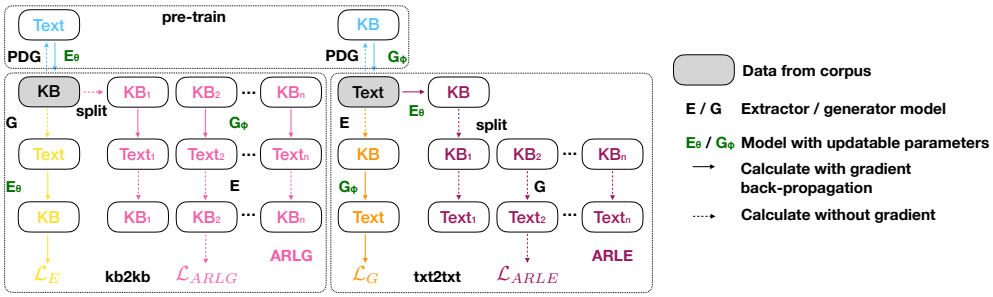

Figure 2: The extractor-generator dual (EGD) framework. It contains three processes namely a pre-train process, a kb2kb process and a txt2txt process.

information narration (Sun et al., 2018). However, as indicated in Hoang et al. (2018), the dual learning approach is not easy to train. Moreover, these methods still need some aligned data to pre-train the weak model. Another line of research proposes to use some extra annotations instead of using aligned data. Lample et al. (2018a,b) propose to train an unsupervised NMT system based on few annotated word pairs (Conneau et al., 2018). Luo et al. (2019) propose to generate pseudo data with a rule-based template (Li et al., 2018). However, these models cannot be directly applied in our scenario since our dataset is too complicated to make these annotations. Fu et al. (2020b) propose to utilize topic information from a dynamic topic tracker to solve the dataset insufficiency problem. Cheng et al. (2020) propose to generate better text description for a few entities by exploring the knowledge from KB and distill the useful part. In the field of computer vision, Zhu et al. (2017) propose cycleGAN which uses a cycled training method that transforms the input into another data form and then transforms it back, minimizing the recover loss. The method works well in the image domain but has some problems in text generation considering the non-differentiable discrete layer. We follow the ideas of cycleGAN to train the whole model without supervised data and adopt the RL method proposed in dual learning methods.

Reinforcement Learning (RL) has been utilized to solve the infeasibility of backpropagation through discrete tokens layer. Li et al. (2016) propose to use RL to focus on the long term target and thus improve the performance. Yu et al. (2017) propose to use the RL in generative adversarial networks to solve the discrete tokens problem. He et al. (2016); Sun et al. (2018) propose to use RL in dual training. As far as we know, no studies of RL have been conducted for KB triples in which the reward is different for each triple considering multiple-triple problem.

3 Method

3.1 Problem Definition

Formally, we denote the KB corpus as $\mathcal{K} = \{K_i|\forall i\}$ in which $K_i = [k_1^{(i)}, k_2^{(i)}, \cdots, k_{n_i}^{(i)}]$ is the ith KB triple list containing n_i triples. $k_j^{(i)} = (h_j^{(i)}, r_j^{(i)}, t_j^{(i)})$ represents the jth KB triple in K_i containing the head, relation and tail entity respectively. We denote the texts corpus as $\mathcal{T} = \{T_i|\forall i\}$ in which $T_i = [t_1^{(i)}, t_2^{(i)}, \cdots, t_{n_i}^{(i)}]$ is the ith sentence and $t_j^{(i)}$ is the jth word in the sentence. In our problem, we are only given a collection of KB triples $\mathcal{K}_t \subset \mathcal{K}$ and a collection of text $\mathcal{T}_t \subset \mathcal{T}$ without any alignment information between them. The ultimate goal is to train a model that generates the corresponding text in $\mathcal{T}$ describing the given triple list from $\mathcal{K}$.

3.2 Extractor-Generator Dual Framework

Our proposed Extractor-Generator Dual (EGD) framework is composed of a generator G and an extractor E that translate data in one form to another. We denote all trainable parameters in E and G as θ and ϕ, respectively. The generator generates text representation for each KB triple as $T' = G(K), K \in \mathcal{K}, T' \in \mathcal{T}$ while the extractor extracts KB triples from raw text as $K' = E(T), T \in \mathcal{T}, K' \in \mathcal{K}$. Our EGD framework is trained in an unsupervised manner and it contains three processes, as shown in Fig. 2. The first process is a pre-train process in which both E and G are trained with the pseudo data generated by the pseudo generator. The second process is the kb2kb process which generates description text based on the given KB triples with G and then recovers the KB triples from the generated text with E. The

third process is called txt2txt which extracts KB triples from the given text with E and then recovers the text from the generated KB triples with G. In order to overcome the multiple-mapping problem, we propose a novel allocated reinforcement learning component in kb2kb and txt2txt, respectively.

The EGD framework firstly pre-trains the extractor and generator with the data generated by the pseudo data generator (PDG). For the text corpus $\mathcal{T}_t$, we generate corresponding pseudo KB triples as $\mathcal{K}'_t = \{K = P_K(T)|\forall T \in \mathcal{T}_t\}$, in which P_K is the pseudo KB generator. We pre-train the generator G to transform $K \in \mathcal{K}'_t$ to $T \in \mathcal{T}_t$. Similarly, we generate pseudo text as $\mathcal{T}'_t = \{T = P_T(K)|\forall K \in \mathcal{K}_t\}$, in which P_T is the pseudo text generator. Then, we train the extractor to transform $T \in \mathcal{T}'_t$ to $K \in \mathcal{K}_t$. After G and E have been pre-trained, the kb2kb process and the txt2txt process are conducted alternately to further improve the performance.

In the kb2kb process, the input KB triples are firstly flattened and concatenated one by one as $K = [k_1, k_2, \cdots, k_{n_k}] = [w_1, w_2, \cdots, w_{n_w}]$ in which k_i is the ith triple in K while w_i denotes the ith words in the concatenated word list. n_k is the number of triples while n_w is the number of the words. K is then sent into the generator G to get a text description $T_m = [t_1, t_2, \cdots, t_{n_t}]$, where t_i is the ith word in the sentence T_m and n_t is the length of T_m. Afterwards, The extractor takes the sentences T_m as input and outputs the triple sequence $K' = [w'_1, w'_2, \cdots, w'_{n'_w}]$, in which w'_i is the ith word in K' while n'_w is the length of K'. The target is to make K' as close to K as possible. Therefore, in the training step, the loss function for the extractor is defined as the negative log probability of each word in K:

$$\mathcal{L}_E = -\sum_{i=1}^{n_w} \log p_\theta(w'_i = w_i|T_m, w_1, \cdots, w_{i-1}).$$

We can also use the output to improve the generator. Since T_m is discrete, the gradient cannot be passed to the generator as the cycleGAN (Zhu et al., 2017) does. To tackle this problem, we propose an Allocated Reinforcement Learning for Generator (ARLG) component to utilize the extractor's result to optimize the generator. Different rewards are allocated to different parts of the generator output. The gradient for the generator is denoted as $\nabla_\phi \mathcal{L}_{ARLG}$ which will be introduced in the later section.

In the txt2txt process, the input text $T = [t_1, t_2, \cdots, t_{n_t}]$ is transformed into its KB representation $K_m = [k_1, k_2, \cdots, k_{n_m}]$ by the extractor E. K_m is then transformed to $T' = [t'_1, t'_2, \cdots, t'_{n_t}]$ by the generator and the loss is defined as:

$$\mathcal{L}_G = -\sum_{i=1}^{n_t} \log p_\phi(t'_i = t_i|K_m, t_1, \cdots, t_{i-1}).$$

Similarly, we also propose an Allocated Reinforcement Learning for Extractor (ARLE) to utilize the generator's result to optimize the extractor. Different rewards are allocated to different parts of the extractor output. Let the gradient for the extractor be denoted as $\nabla_\theta \mathcal{L}_{ARLE}$. The final gradient for extractor's parameters θ is formulated as $\nabla_\theta \mathcal{L}_E + \nabla_\theta \mathcal{L}_{ARLE}$ while the gradient for generator's parameters ϕ is $\nabla_\phi \mathcal{L}_G + \nabla_\phi \mathcal{L}_{ARLG}$. We use the Adam (Kingma and Ba, 2014) as the optimizer to optimize all the parameters.

3.3 Background of Transformer

The extractor and the generator are both backboned by the prevalent Transformer (Vaswani et al., 2017) model, which is a variant of the seq-to-seq model. It takes a sequence as input and generates another sequence as output. The Transformer model contains two parts, namely an encoder and a decoder. Both of them are built with several attention layers. We refer readers to the original paper (Vaswani et al., 2017) for more details.

3.4 Pseudo Data Generator

To handle the cold start problem, we propose a novel pseudo data generator (PDG) to generate pseudo data. It contains two components, namely a pseudo text generator and a pseudo KB generator.

Pseudo Text Generator generates pseudo text for each KB and forms a pseudo supervised training data for pre-training the extractor and thus solving the cold start problem. We compute a statistics of the word count in the training set $\mathcal{T}_t$ and calculate the empirical distribution for each word as:

$$p(w) = \frac{\#w}{\sum_{w' \in \mathcal{T}_t} \#w'},$$

where $\#w$ stands for the total word count for w in $\mathcal{T}_t$. For a list of KB triples $K = [k_1, k_2, \cdots, k_{n_k}] = [h_1, r_1, t_1, h_2, r_2, t_2, \cdots, h_{n_k}, r_{n_k}, t_{n_k}]$, we firstly sample head entities and tail entities as $K_s = [h_1, t_1, h_2, t_2, \cdots; h_n, t_n]$. The final

sequence is generated by sampling from both K_s and $p(w)$. When generating each word $\tilde{T}_i$, a random number generator is used to generate a random number r_i uniformly. r_i is used to compare with a threshold parameter α. If $r_i > \alpha$, $\tilde{T}_i$ is sampled with the word distribution $p(w)$, otherwise, it is sampled form the next token in K_s. This process can be expressed mathematically as:

$$\tilde{T}_i = \begin{cases} w \sim p(w) & r_i > \alpha \\ K_s[1 + \sum_{j=1}^{i-1} \mathbb{1}(\tilde{T}_j \in K_s)] & \text{otherwise} \end{cases},$$

in which $\mathbb{1}(C) = 1$ if condition C is true and 0 otherwise. $\tilde{T}_j \in K_s$ indicates whether the word $\tilde{T}_j$ is sampled from K_s. This pseudo text data is used to solve the cold start problem when training the extractor.

Pseudo KB Generator generates pseudo KB triples for each text and form a pseudo supervised training data. This data is used to solve the cold start problem when pre-training the generator. Similar with the work of Freitag and Roy (2018), for an input sequence T we randomly remove words in the input text with a probability β_1 and sample new words by sampling words from a distribution with a probability β_2. The generated sequence $\tilde{K}$ is the pseudo KB sequence for each text. Similar to the Pseudo Text Generator, we randomly add some words by sampling from the distribution $p(w)$. We do not use the probability calculated from $\mathcal{K}_t$ since it may sample some wrong relations or wrong entity names which undermines the performance. Mathematically, it can be expressed as:

$$\tilde{K}_i = \begin{cases} w \sim p(w) & r_i < \beta_2 \\ T_s[1 + \sum_{j=1}^{i-1} \mathbb{1}(\tilde{K}_j \in T_s)] & \text{otherwise} \end{cases},$$

in which $T_s = s(T)$ and $s(\cdot)$ is a sample function defined as:

$$s(T) = \begin{cases} T & \|T\| = 0 \\ [T_1; s(T_{2:\|T\|})] & r < \beta_1, \|T\| \neq 0 \\ s(T_{2:\|T\|}) & \text{otherwise} \end{cases},$$

where $\|T\|$ denotes the length of the sequence T while $T_{2:\|T\|}$ stands for the sub-sequence from the second to the last of T.

3.5 Allocated Reinforcement Learning

Traditional reinforcement learning for sequence generation calculates a reward for the whole sequence (He et al., 2016; Hoang et al., 2018; Keneshloo et al., 2018) and uses the policy gradient (Sutton et al., 2000) algorithm to optimize the parameters. It suffers from the multiple-triple problem as discussed above. We propose an allocated reinforcement learning method to allocate different rewards for different KB triples and thus alleviate this problem. In the kb2kb process, the RL model is called the Allocated Reinforcement Learning for Generator (ARLG) since it optimizes the parameters in the generator while in the txt2txt process, it is called Allocated Reinforcement Learning for Extractor (ARLE) accordingly.

ARLE is shown in Fig. 2. The main idea is to recover and evaluate the KB triples separately which inherently has the following benefits: 1) Each triple is given a distinct reward as discussed above; 2) Traditional RL is more likely to ignore some triples (e.g., 3rd triple in Fig. 1) since it handles several triples at once while our method alleviates such problem by handling triples one by one. It firstly sends the input text $T = [t_1, t_2, \cdots, t_{n_t}]$ into the extractor and get the extracted triples: $K_m = E(T) = [k_m^{(1)}, k_m^{(2)}, \cdots, k_m^{(n_k)}]$. The corresponding probability for each token is denoted as $p_j^{(i)}$, in which i denotes the ith triple and j denotes the jth word in the triple. Afterwards, the generator is applied on each triple in K_m to recover the corresponding text, which denotes as: $T' = [G(k_m^{(1)}), G(k_m^{(2)}), \cdots, G(k_m^{(n_k)})] = [t'_1, t'_2, \cdots, t'_{n_k}]$. We calculate the reward for each $k_m^{(i)}$ as the recall for each corresponding t'_i referring to T:

$$R(k_m^{(i)}) = \frac{\sum_{j=1}^{\|t'_i\|} \mathbb{1}(t_i'^{(j)} \in T)}{\|t'_i\|},$$

in which $\|t'_i\|$ denotes the length of t'_i and $t_i'^{(j)}$ is the jth word in t'_i. The reward for each sentence in K_m is denoted as: $R_e = [R(k_m^{(1)}), R(k_m^{(2)}), \cdots, R(k_m^{(n_k)})]$. Different from the traditional policy gradient algorithm (Sutton et al., 2000), our RL uses a different reward for each generated triple. The gradient is calculated as:

$$\nabla_\theta \mathcal{L}_{ARLE} = -\mathbb{E}[\sum_{i=1}^{n_k} R(k_m^{(i)}) \sum_{j=1}^{\|k'_i\|} \nabla_\theta \log p_j^{(i)}].$$

Since the RL model only guides the model with some reward scores which is only one aspect of the result. It misleads the model into generating some sequences which have a high reward while actually perform worse. To prevent this, we propose to conduct the gradient descent together with the kb2kb process simultaneously in which the extractor is trained with a supervised sequence.

ARLG is applied in the kb2kb process. The input KB triples is firstly splitted into n_k triples $K = [k_1, k_2, \cdots, k_{n_k}]$ which is then sent into the generator separately and get the corresponding description sentences: $T_m = [G(k_1), G(k_2), \cdots, G(k_{n_k})] = [t_m^{(1)}, t_m^{(2)}, \cdots, t_m^{(n_k)}]$. The corresponding probability for the jth word in the ith sentence is denoted as $p_j^{(i)}$. Afterwards, the text is sent into the extractor to recover the input KB triple for each $t_m^{(i)}$: $K' = [E(t_m^{(1)}), E(t_m^{(2)}), \cdots, E(t_m^{(n_k)})] = [k'_1, k'_2, \cdots, k'_{n_k}]$. We calculate the reward for each $t_m^{(i)}$ as the precision for each corresponding k'_i referring to k_i in K:

$$P(t_m^{(i)}) = \frac{\sum_{j=1}^{\|k'_i\|} \mathbb{1}(k_i'^{(j)} \in k_i)}{\|k'_i\|},$$

in which $\|k'_i\|$ denotes the total word number count of k'_i. The reward for each sentence in T_m is denoted as: $R_g = [P(t_m^{(1)}), P(t_m^{(2)}), \cdots, P(t_m^{(n_k)})]$. We use RL to maximize the expected reward for each KB triple $t_m^{(i)}$ with corresponding reward. The gradient is:

$$\nabla_\phi \mathcal{L}_{ARLG} = -\mathbb{E}[\sum_{i=1}^{n_k} P(t_m^{(i)}) \sum_{j=1}^{\|t'_i\|} \nabla_\theta \log p_j^{(i)}].$$

Similar to ARLE, we also train the model with the txt2txt process to give a targeted sequence to guide the training together with the reward score.

4 Experiments

4.1 Dataset

We adopt the WebNLG v2 dataset (Gardent et al., 2017a)[1]. It samples KB triples from DBpedia and annotates corresponding texts by crowdsourcing. In order to show that our model can work under the unsupervised setting, we split the original dataset into two parts, namely the KB part and the text part. We do not assume any alignment between KB and text. Table 1 shows the statistics of instances with different number of triples. In this dataset, one sentence can be mapped to at most seven triples. We use the same dev and test set as the original WebNLG. The training set has 34,352 samples in total while the dev set and the test set have 4,316 and 4,224 samples respectively. It can be observed that there are 78.2% sentences mapped with multiple-triple.

#triples	1	2	3	4	5	6	7	Total
train	7,429	6,717	7,377	6,888	4,982	488	471	34,352
dev	924	842	919	877	632	64	58	4,316
test	931	831	903	838	608	58	55	4,224

Table 1: Statistics for the dataset. The number of instances with different number of triples are listed.

4.2 Comparison Models

We compare our model against the following baseline methods:

PDG uses the Pseudo Data Generator to generate the pseudo data for pre-training both extractor and generator. PDG does not conduct the subsequent dual learning process and thus illustrates the capability of PDG.

DL uses the dual learning process proposed in He et al. (2016); Zhu et al. (2017). It is fine-tuned on the PDG model and iterates alternatively between txt2txt and kb2kb processes. Here, we do not use any reinforcement learning component.

DL-RL1 uses the dual learning process together with an RL component. It is similar to the dual learning method proposed in He et al. (2016); Zhu et al. (2017). We use the PDG's data to train the weak model. It uses the log-likelihood of the recover process's output sequence as the reward.

DL-RL2 follows the settings of Sun et al. (2018). Different from DL-RL1, this model uses the ROUGE$_L$ (Lin, 2004) score of the recovered sequence instead of using the log-likelihood as the reward.

SEG is a Supervised Extractor-Generator using the original setting of WebNLG for both generator and extractor. It utilizes all the alignment information between KB and text and thus provides an upper bound for our experiment.

4.3 Experimental settings

We evaluate the performances of the generator and the extractor with several metrics including BLEU (Papineni et al., 2002), NIST (Dodding-

[1]https://gitlab.com/shimorina/webnlg-dataset

	Generator					Extractor							
	BLEU	NIST	METEOR	ROUGE$_L$	CIDEr	BLEU	NIST	METEOR	ROUGE$_L$	CIDEr	Precision	Recall	F1
PDG	0.322	7.06	0.349	0.505	2.63	0.489	6.01	0.351	0.618	3.97	0.635	0.465	0.510
DL	0.352	7.71	0.347	0.528	2.96	0.735	10.4	0.502	0.743	5.67	0.644	0.691	0.646
DL-RL1	0.356	7.73	0.350	0.532	3.00	0.760	10.8	0.501	0.755	5.92	0.670	0.687	0.658
DL-RL2	0.356	7.75	0.350	0.533	2.99	0.757	10.7	**0.503**	0.755	5.90	0.668	**0.691**	0.659
EGD	**0.369**	**7.77**	**0.364**	**0.541**	**3.13**	**0.775**	**11.1**	**0.503**	**0.772**	**6.25**	**0.704**	**0.691**	**0.680**
EGD w/o ARLE	0.351	7.72	0.347	0.529	2.97	0.770	10.9	0.501	0.764	6.11	0.683	0.682	0.665
EGD w/o ARLG	0.353	7.77	0.348	0.531	2.99	0.729	10.4	0.505	0.746	5.61	0.639	0.695	0.645
EGD w/o PDG	0.010	0.82	0.037	0.119	0.02	0.020	0.42	0.026	0.042	0.08	0.011	0.008	0.007
SEG	0.406	8.31	0.385	0.585	3.66	0.848	11.8	0.595	0.867	7.43	0.783	0.830	0.796

Table 2: Results for generator (left) and extractor (right), which are evaluated with generation metrics. For the extractor, precision, recall, and F1 scores are also calculated at triple's level. The performances of our EGD method without different components and the supervised method SEG are shown in the bottom.

Ratio	Generator		Extractor	
	BLEU	ROUGE$_L$	BLEU	ROUGE$_L$
0.10	0.235	0.439	0.335	0.557
0.15	0.281	0.49	0.655	0.708
0.20	0.308	0.506	0.746	0.757
0.25	0.347	0.524	0.71	0.764
PDG	0.322	0.505	0.489	0.618

Table 3: Compare our PDG framework with semi-supervised models at different labeling ratios.

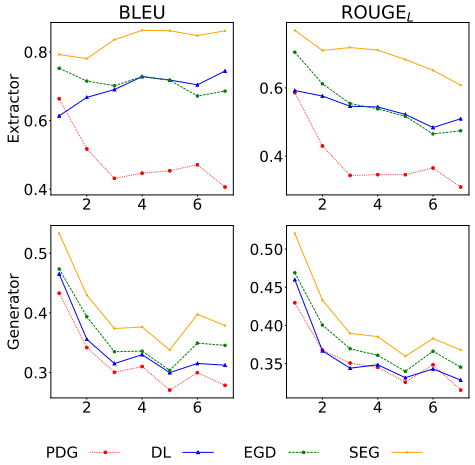

Figure 3: The influence of KB triples count. The x-axis represents the KB triples count while the y-axis represents the scores.

ton, 2002), METEOR (Banerjee and Lavie, 2005), ROUGE$_L$ (Lin, 2004), and CIDEr (Vedantam et al., 2015). These metrics are calculated with the evaluation code provided in Novikova et al. (2017). Moreover, we also evaluate the performance of the extractor with precision, recall, and F1 scores (Manning et al., 2010). In PDG, we set $\alpha = 0.8, \beta_1 = 0.2, \beta_2 = 0.6$. We firstly pre-train the extractor and the generator in the PDG model with the data generated by PDG until convergence. All other models are fine-tuned on the PDG model. For the DL model, we train the generator for 5 steps with the txt2txt process and train the extractor with the kb2kb process for another 5 steps with the new generator. We iterate this process 10 times. For all transformers, we set clip norm to 1.0, label smoothing to 0.1, and dropout to 0.3. We use Adam (Kingma and Ba, 2014) as our optimizer and set the learning rate for the extractor to 2e-4 and generator to 5e-4. All hyper-parameters are tuned on the dev dataset with grid search.

4.4 Experimental Results

The performances of our KB-to-text generator and triple extractor are shown in the left and right of Table 2 respectively. Both generator and extractor of our model outperform all baseline models significantly and consistently. The comparison between our EGD model and the supervised SEG model indicates that our unsupervised EGD model is close to the bound of the supervised methods. Compared with the PDG model, our EGD model has a much better performance with the dual learning framework and the ARL component. Moreover, Our EGD model outperforms the DL-RL1 and DL-RL2 model, which indicates that our proposed ARL component can handle the multiple-triple problem between triples and texts. In the traditional RL models, the reward is the same for a whole sequence including all the triples while in our ARL model, the reward is calculated for several subparts of the sequence, which is more accurate and effective. By comparing PDG with SEG, we found that the model trained with our proposed pseudo data generator (PDG)'s output achieves acceptable results. It indicates that using the PDG's output is a feasible alternative to initialize the model and can handle the cold start problem.

Ablation Study. We also conduct some ablation studies to show that each component contributes

	Extractor	Generator
Gold	(1634 : The Ram Rebellion, mediaType, E - book) (1634 : The Ram Rebellion, author, Virginia DeMarce)	Virginia DeMarce is the author of 1634 : The Ram Rebellion , which can be found as an e - book .
SEG	(1634 : The Ram Rebellion, mediaType, E - book) (1634 : The Ram Rebellion, author, Virginia DeMarce)	1634 : The Ram Rebellion was written by Virginia DeMarce and has the ISBN number 1 - 4165 - 2060 - 0 .
PDG	(1634 : The Ram Rebellion, mediaType, E - book)	1634 : The Ram Rebellion was followed by 1634 : The Galileo Affair and its author is Virginia DeMarce .
DL	(1634 : The Ram Rebellion, mediaType, E - book) (1634 : The Ram Rebellion, author, Virginia DeMarce) (1634 : The Ram Rebellion, ISBN number, 1 - 4165 - 2060 - 0)	1634 : The Ram Rebellion is available as an E - Book .
EGD	(1634 : The Ram Rebellion, mediaType, E - book) (1634 : The Ram Rebellion, author, Virginia DeMarce)	Virginia DeMarce is the author of 1634 : The Ram Rebellion , currently in print .

Table 4: Case study. The input KB and text are listed in the first row.

to the final performance. The results are shown at the bottom part of Table 2. By comparing the model EGD w/o ARLE and EGD w/o ARLG with the EGD model, we can see that both the ARLE and ARLG components are effective to handle the multiple-triple problem and help improve the performance. It is interesting to see that the result of EGD w/o PDG is extremely poor showing the importance of our PDG component. The EGD w/o PDG removes the pre-train stage with the pseudo data generator and conducts the iterations between txt2txt and kb2kb directly. Without PDG, we observe that the models trend to learn some "own language" without a good initialization which is incomprehensible to human.

The Influence of the KB triples Number. We analyze the influence of the KB triples' number on the performance. The results are shown in Fig. 3. As expected, the SEG model performs the best over all numbers since it is fully supervised. The PDG model performs the worst since it only uses pseudo data to train. The DL model improves significantly comparing with the PDG model over all numbers, especially in the extractor model. It shows that using dual learning's iteration approach does improve the model of training solely based on PDG's data. Our proposed EGD model outperforms the DL model and the PDG model. This shows that the ARL model does help to give more information to train the model. Nearly all generators' scores decrease as the number increases. This is because if the sequence is long, it has more ways to express those triples which may be different from the gold standard sentence. However, when extracting triples from the text, it only has one correct way and thus the extractor's scores are similar in all lengths.

Error Analysis. We conduct an error analysis experiment for the top 20 mentioned relations in the extractor which is shown in Fig. 4. We focus on two kinds of errors. The first kind of error is called "false negative" which means when extracting, some correct triples are ignored. The second kind of error is called "false positive" which means that the extractor generates some incorrect triples that the text does not mention. It can be observed that the "false negative" problem is much more severe than the "false positive" problem for the PDG model, while the DL model and the EGD model alleviate this problem a lot. This reason is that the pseudo text data is made by sampling entities in KB ignoring relation information. Iterating alternately between txt2txt and kb2 solves the problem since the missing information is supplemented. It can also be observed that when comparing with the DL model, our EGD model mainly solves the "false positive" problem. The reason is that the RL can penalize the wrong generated triples but cannot give specific guidance on which missing triples the model should generate.

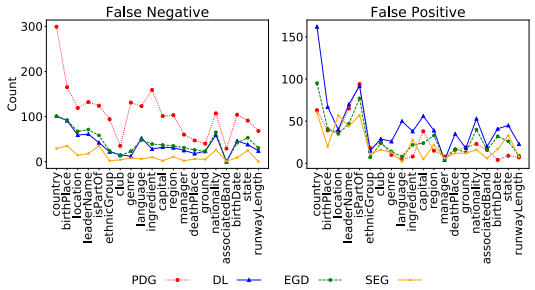

Figure 4: Error analysis for top 20 mentioned relations.

Comparison with Semi-Supervised Learning. To measure the quality of the initialization via PDG, we compare our PDG method against the semi-supervised learning method. We sample labeled data from the original dataset with different ratios to train the models and compare the results with the PDG model. The result is shown in Table 3. It can be concluded from the result that training the

extractor with the PDG's data outperforms training with 10% aligned data and it also outperforms 20% aligned data for the generator. It shows that our PDG component does provide usable data and it can be boosted a lot in the subsequent dual iteration process.

Case Study. Table 4 shows a case study for 4 models. For the extractor, the input is "*Virginia DeMarce is the author of 1634 : The Ram Rebellion , which can be found as an e - book .*". For the generator, the input is "*(1634 : The Ram Rebellion, mediaType, E - book) (1634 : The Ram Rebellion, author, Virginia DeMarce)*". It can be observed that for the PDG model, it omits the second triple. It also shows that the PDG model has a severe false negative problem which has been mentioned in the error analysis sub-section. The DL model alleviates this problem but it introduces more triples causing the false positive problem. Our EGD model solves the false positive problem by the RL component. All models make some mistakes in the generation process including the supervised SEG model. The result of the generator shows that it is more difficult to generate a sequence than extracting triples.

5 Conclusions

We propose a new challenging task, namely, unsupervised KB-to-text generation. To solve this task, we propose an extractor-generator dual framework which exploits the inverse relationship between the KB-to-text generation task and the auxiliary triple extraction task. To handle the cold start problem and the multiple-triple problem respectively, we propose a novel pseudo data generator and an allocated reinforcement learning component. Experimental results show that our proposed method successfully resolves the observed problems and outperforms all the baseline models.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, pages 722–735.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. Seqˆ3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression. 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 673–681.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 51–64, Florence, Italy. Association for Computational Linguistics.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Liying Cheng, Dekun Wu, Lidong Bing, Yan Zhang, Zhanming Jie, Wei Lu, and Luo Si. 2020. Entdesc: Entity description generation by exploring-knowledge graph. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145.

Markus Freitag and Scott Roy. 2018. Unsupervised natural language generation with denoising autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3922–3929.

Zihao Fu, Lidong Bing, and Wai Lam. 2020a. Open domain event text generation. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7748–7755.

Zihao Fu, Lidong Bing, Wai Lam, and Shoaib Jameel. 2020b. Dynamic topic tracker for kb-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics: Technical Papers (COLING)*.

Zihao Fu, Bei Shi, Wai Lam, Lidong Bing, and Zhiyuan Liu. 2020c. Partially-aligned data-to-text generation with distant supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24.

Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K Reddy. 2018. Deep reinforcement learning for sequence to sequence models. *arXiv preprint arXiv:1805.09461*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1203–1213.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1865–1874.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Workshop on Text Summarization Branches Out*.

Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. A dual reinforcement learning framework for unsupervised text style transfer. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Mingming Sun, Xu Li, and Ping Li. 2018. Logician and orator: Learning from the duality between language and knowledge in open domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2119–2130.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2253–2263.

Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.