# Partially-Aligned Data-to-Text Generation with Distant Supervision

**Zihao Fu**[1], **Bei Shi**[2], **Wai Lam**[1], **Lidong Bing**[3], **Zhiyuan Liu**[4]
[1]Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong
[2]Tencent AI Lab [3]DAMO Academy, Alibaba Group
[4]State Key Lab on Intelligent Technology and Systems
Department of Computer Science and Technology
Institute for Artificial Intelligence, Tsinghua Univeristy
zhfu@se.cuhk.edu.hk; beishi@tencent.com; wlam@se.cuhk.edu.hk;
l.bing@alibaba-inc.com; liuzy@tsinghua.edu.cn

## Abstract

The Data-to-Text task aims to generate human-readable text for describing some given structured data enabling more interpretability. However, the typical generation task is confined to a few particular domains since it requires well-aligned data which is difficult and expensive to obtain. Using partially-aligned data is an alternative way of solving the dataset scarcity problem. This kind of data is much easier to obtain since it can be produced automatically. However, using this kind of data induces the over-generation problem posing difficulties for existing models, which tends to add unrelated excerpts during the generation procedure. In order to effectively utilize automatically annotated partially-aligned datasets, we extend the traditional generation task to a refined task called Partially-Aligned Data-to-Text Generation (PADTG) which is more practical since it utilizes automatically annotated data for training and thus considerably expands the application domains. To tackle this new task, we propose a novel distant supervision generation framework. It firstly estimates the input data's supportiveness for each target word with an estimator and then applies a supportiveness adaptor and a rebalanced beam search to harness the over-generation problem in the training and generation phases respectively. We also contribute a partially-aligned dataset [1] by sampling sentences from Wikipedia and automatically extracting corresponding KB triples for each sentence from Wikidata. The experimental results show that our framework outperforms all baseline models as well as verify the feasibility of utilizing partially-aligned data.

## 1 Introduction

The Data-to-Text generation task focuses on generating human-readable text corresponding to some given structured data. For example, given the input knowledge base (KB) triple ⟨Company of Heroes, developer, Relic Entertainment⟩, the aim is to generate a text description such as "*Company of Heroes is developed by Relic Entertainment.*". In recent years, many works have been proposed to give impetus to the Data-to-Text generation task. For instance, Gardent et al. (2017a; 2017b) proposed the WebNLG task aiming at generating description text of the given KB triples. Novikova et al. (2017) proposed the E2E task aiming at generating restaurant reviews according to the given restaurant attributes. Lebret et al. (2016) proposed the WikiBio task in which the biography of each person is generated according to the given Wikipedia infobox.

These typical data-to-text generation tasks are confined to a few particular domains since it requires well-aligned data and text pairs which are difficult and costly to obtain. Specifically, it is required that each input data provides exactly the same information with the target text. This requirement makes the dataset difficult to build and confines the task to particular domains where such kind of data (WikiBio, E2E) or human-labeled data (WebNLG) are available. Using partially-aligned data is an alternative way of solving the dataset scarcity problem. Partially-aligned data do not re-

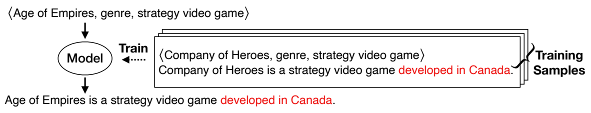
⟨Age of Empires, genre, strategy video game⟩
Model
Train
⟨Company of Heroes, genre, strategy video game⟩
Company of Heroes is a strategy video game developed in Canada.
Training Samples
Age of Empires is a strategy video game developed in Canada.

Figure 1: Illustration of the over-generation problem in the partially-aligned data-to-text generation task. In the training set, there is no KB triple corresponding to the text "*developed in Canada*". The model is likely to bind the text to existing triples incorrectly. As a result, during the testing or operational stage, the model is likely to overly generate this kind of excerpt for similar triples.

[1]The data and source code of this paper can be obtained from `https://github.com/fuzihaofzh/distant_supervision_nlg`

quire that each part in the text is exactly aligned with a particular input KB triple. This kind of data is much easier to obtain with automatic methods. Consequently, it can handle much broader kinds of domains. However, it induces the over-generation problem[2]. As shown in Fig. 1, some parts ("*developed in Canada*") in the generated text for $\langle$Age of Empires, genre, strategy video game$\rangle$ have not been mentioned in the input KB triple. Essentially, it is because in the training set, such unrelated text exists in some training samples. During the training, it misleads the model to bind the text "*developed in Canada*" to some irrelevant KB triples. When similar triples exist in the testing, it is prone to adding some over-generated text which is actually unrelated to the given input data. Current generation models fail to be trained on such partially-aligned data due to lacking the tolerance of the over-generation problem.

In order to effectively utilize automatically annotated partially-aligned datasets for handling more domains, we extend the traditional generation task to a refined task called Partially-Aligned Data-to-Text Generation (PADTG). Like the traditional task, the PADTG task also requires generating text with respect to the given input data. However, for the training data, we only require that the given structured data contains partial information of the corresponding text. This task is more practical since it utilizes the partially-aligned data for training and thus considerably expands the application domains. However, due to such data's nature, successfully suppressing the over-generation problem is the critical point for proposing an effective model.

We propose a Distant Supervision Generation (DSG) framework to tackle the PADTG task. Our framework can deal with the challenging over-generation problem when training on the partially-aligned data. It firstly trains an estimator to calculate each word's supportiveness in the target sentence with respect to the input data, i.e. how likely the word is conveyed by the input triples. Then the framework employs a sequence-to-sequence (S2S) neural model to encode the input data and generates the description sentence accordingly. In the training procedure, a supportiveness adaptor is used to adapt the estimated supportiveness into the loss function while in the generation procedure, a rebalanced beam search is used to generate text augmented with the supportiveness scores.

To prepare the partially-aligned data, we build a new dataset called WITA from text sources, namely, Wikipedia and Wikidata. We propose a novel KB extractor to extract KB triples given a piece of text sampled from Wikipedia. The KB extractor firstly detects named entities with an entity detector. The triple retriever queries the Wikidata database to find the most matching triples corresponding to these entities. We filter the results with a matching score to remove unextractable sentences.

Our contributions can be summarized as follows. (1) We propose a new task, namely, partially-aligned Data-to-Text generation, which is more practical and extensible to more domains. (2) We propose a distant supervision generation framework that can tackle the challenges of the new task including the over-generation problem. (3) We contribute a sizeable partially-aligned dataset suitable for this task.

## 2 Method

### 2.1 Overview

Formally, we denote the input KB triples as $K = [\langle h_1, r_1, t_1\rangle, \cdots, \langle h_n, r_n, t_n\rangle]$, where $h_i, r_i, t_i$ represent the $i$th head, relation, and tail respectively while $n$ is the number of triples. The corresponding text is denoted as $T = [w_1, \cdots, w_m]$, in which $w_i$ is the $i$th word in $T$ and $m$ is the sentence length. It should be noted that, in the task of Partially-Aligned Data-to-Text Generation (PADTG), $T$ contains some information that $K$ does not have. The target of the task is to train a model that generates text $T'$ that exactly describes the KB triples in $K$.

Our proposed Distant Supervision Generation (DSG) framework contains four components, namely a Supportiveness Estimator (SE), a Sequence-to-Sequence Generator (S2SG), a Supportiveness Adaptor (SA), and a Rebalanced Beam Search (RBS). As illustrated in Fig. 2, in the SE training procedure, we first pre-train the SE component to estimate a supportiveness vector $s \in \mathbb{R}^m$ indicating whether each target word $w_i \in T$ is describing the input triples in $K$. It adopts the self-supervised mechanism that trains the model to maximize the margin between the target words' scores and negative sampled words' scores. Then, the pre-trained SE component is utilized to estimate a supportiveness vector $s$ in both S2SG Training and S2SG Generation. In the S2SG training procedure, the S2SG model firstly calculates the generation

[2]Note that omission problem also happens in the Data-to-Text generation task, but it is not a major problem.

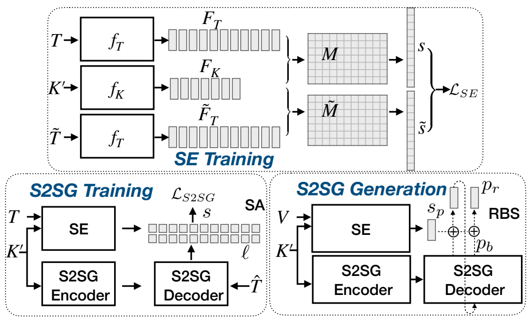

Figure 2: Distant Supervision Generation Framework.

loss $\ell$. Then, SA combines $\ell$ with $s$ to get a refined loss in which the loss is diminished if one target word has lower supportiveness. In the S2SG generation procedure, the RBS component combines $s$ with the probability distribution of candidate words to obtain a better generation result.

## 2.2 Supportiveness Estimator

We concatenate the input KB triples word-by-word as $K' = [w_1^{h_1}, \cdots, w_{|h_1|}^{h_1}, w_1^{r_1}, \cdots, w_{|r_1|}^{r_1}, w_1^{t_1}, \cdots, w_{|t_1|}^{t_1}, \text{KBSEP}, w_1^{h_2} \cdots\cdots, w_{|t_n|}^{t_n}]$, in which $h_j, r_j, t_j$ is the head, relation and tail entity of the $j$th triple. $w_i^{h_j}$ represents the $i$th word in the $j$th KB triple's head entity. $|h_j|$ stands for the word count of the $j$th head entity. KBSEP is a seperator between each triple.

**Feature Extraction**. In SE, a feature extraction component $f_K$ is utilized to extract features for each word denoted as $F_K = f_K(K')$, in which $F_K \in \mathbb{R}^{d\times|K'|}$ is the extracted feature matrix for $K'$. $d$ is the embedding dimension and $|K'|$ is the length of $K'$. Specifically, in the feature extraction component, $K'$ is firstly embedded with an embedding layer as $K_1 = \text{emb}(K')$, $K_1 \in \mathbb{R}^{d\times|K'|}$. Then $K_1$ is sent into a normalization layer (Ba et al., 2016) as $K_2 = \text{NL}(K_1)$, $K_2 \in \mathbb{R}^{d\times|K'|}$. NL is defined as

$$\text{NL}(x) = \frac{\gamma(x - \mathbb{E}(x))}{\sqrt{\text{Var}(x) + \epsilon}} + \beta,$$

in which $\mathbb{E}$ and Var are mean and variance of the input $x$ while $\gamma$ and $\beta$ are learnable parameters. $\epsilon$ is a small constant which is usually set to $1.0e-5$. $K_2$ is then sent into a combination of linear feedforward layers and a ReLU layer as $K_3 = \text{FW}_2(\text{ReLU}(\text{FW}_1(K_2)))$, $K_3 \in \mathbb{R}^{d\times|K'|}$, where $\text{FW}_2$ and $\text{FW}_1$ are linear feedforward layer while ReLU stands for the ReLU layer. Afterwards, the features representation is calculated as $F_K = \text{NL}(K_3)$. Similarly, the features for each word in the target text is denoted as $F_T = f_T(T)$, $F_T \in \mathbb{R}^{d\times m}$.

**Supportiveness Vector**. We calculate the supporting matrix as $M = F_K^T F_T$, $M \in \mathbb{R}^{|K'|\times m}$, in which $M_{i,j}$ represents the supportiveness of the $i$th word in $K'$ that support for the $j$th word in $T$. The supportiveness score vector is aggregated from $M$ as

$$s_j = \log \sum_{i=1}^{|K'|} \exp(M_{i,j}),$$

where $s_j$ is the $j$th element of the vector $s \in \mathbb{R}^m$ and it stands for input $K$'s supportiveness to the $j$th word.

**Negative Sampling**. In order to prevent the model from giving all words a high supportiveness score, we use the negative sampling method to sample some negative sentences. We denote the empirical distribution of the words in the target text as $P_{\mathcal{T}}$ in which $\mathcal{T}$ is the set of all target sentences. We sample words from $P_{\mathcal{T}}$ while avoiding sampling the same words in T. The sampling procedure can be denoted as $\tilde{w}_i \sim P_{\mathcal{T}}, \tilde{w}_i \notin T$. The negative sample is composed of $\tilde{w}_i$s as $\tilde{T} = [\tilde{w}_1, \cdots, \tilde{w}_m]$, where $\tilde{w}_i$ is the $i$th word in $\tilde{T}$ which has the same length as $T$. The negative sample $\tilde{T}$ will also be fed to the network in the same way as the original target $T$. The supportiveness score vector for the negative sample is denoted as $\tilde{s} \in \mathbb{R}^m$.

**Optimization Target**. The overall loss function consists of a margin loss, a word-consistent loss, and a concentration loss. The **margin loss** is defined as the margin between the supportiveness of the original text and that of the negative sample, which can be written as

$$\mathcal{L}_m = \sum_{i=1}^{\tilde{m}} \sigma(\tilde{s}_i) - \sum_{i=1}^{m} \sigma(s_i),$$

in which $\sigma(x) = 1/(1+e^{-x})$ is a sigmoid function. Minimizing $\mathcal{L}_m$ helps maximize the gap between the positive and the negative samples. The **word-consistent loss** is used to make the supportiveness from the same word in input KB larger than the supportiveness from different words. It is defined as

$$\mathcal{L}_w = -\sum_{i=1}^{m}\sum_{j=1}^{|K'|} \mathbb{1}(T_i = K'_j)[M_{i,j} - \log(\sum_{k=1}^{|K'|} \exp M_{k,j})].$$

It increases the supportiveness $M_{ij}$ if the $i$th word in $T$ and the $j$th word in $K'$ are the same word. The

**concentration loss** is used to avoid one word in $K'$ supporting too many words in $T$. It is denoted as

$$\mathcal{L}_c = \max_i \sum_{j=1}^{m} M_{i,j}.$$

If one word supports too many words, all its corresponding supportiveness will be penalized. The overall loss function is denoted as the weighted sum of these loss functions as

$$\mathcal{L}_{SE} = \mathcal{L}_m + \omega_w \mathcal{L}_w + \omega_c \mathcal{L}_c,$$

in which $\omega_w$ and $\omega_c$ are tunable hyper-parameters.

### 2.3 Sequence-to-Sequence Generator

We use the Transformer (Vaswani et al., 2017) structure as our S2S generator. The Transformer is an attention-based structure and is widely used in many tasks. It contains two major components, namely an encoder and a decoder. All of these components are built with several attention layers. The encoder firstly takes $K'$ as input to generate the feature representation: $G_K = \text{Enc}(K')$. The decoder takes shifted target text $\hat{T}$ (shift last EOS tag to the beginning) as input and get the negative log-likelihood for each word in $T$ as $\ell = \text{Dec}(\hat{T}, G_K)$, $\ell \in \mathbb{R}^{|T|}$, where Dec is the transformer decoder and $|T|$ is the length of $T$. We refer readers to Vaswani et al. (2017) for more technical details.

### 2.4 Supportiveness Adaptor

The supportiveness Adaptor adapts the supportiveness score to S2SG's output. We investigate three methods, namely, Hard Adaptor, Soft Adaptor, and Attention Adaptor.

**Hard Adaptor**. With the supportiveness scores, we can simply remove words that have low supportiveness. For each word $w_i$ in the target sentence $T$, we use a uniform random number generator to generate a random number $r_i \in [0, 1]$. we ignore words in $T$ if $r_i > s_i$ and copy it into $T'$ otherwise. Then, $T'$ is used instead of $T$ in the training procedure.

**Soft Adaptor**. Since the hard adaptor directly removes words, it is easy to omit essential words and make the model generating unreadable text. We propose a soft adaptor to alleviate this issue. We use the original target text $T$ as input. For the output negative log-likelihood loss vector $\ell$, we combine it with the supportiveness vector $s$ to modify the S2SG's loss as

$$\mathcal{L}_{S2SG} = \sum_{i=1}^{m} \ell_i s_i.$$

**Attention Adaptor**. Instead of using SE to estimate the supportiveness vector, attention adaptor directly aggregates the attention matrix as the supportiveness scores in our proposed DSG model. For each target word, it takes its max attention weight on each source word as the supportiveness score. We use maximization to aggregate the scores instead of considering all scores because all attention weights sum up to 1, and thus irrelevant words can also be assigned some attention. Using maximization aggregation avoids such irrelevant words. The supportiveness scores are then utilized in a similar way as the soft adaptor.

### 2.5 Rebalanced Beam Search

In the generation step, the supportiveness scores can also be utilized to help rebalance the final word probability distribution. We make a pseudo target sequence as $T_p = [1, 2, \cdots, |V|]$ which contains all words in vocabulary $V$. The supportiveness score $s_p \in \mathbb{R}^{|V|}$ for all words is calculated as the same procedure in training. In the traditional beam search, it outputs a probability $p_b \in \mathbb{R}^{|V|}$ over the whole vocabulary denoting the possibility for each token in the vocabulary to be chosen as the next word. We rebalance the probability with the supportiveness score vector as $p_r = p_b \cdot s_p^{\alpha}$ where $\alpha$ is a tunable hyper-parameter.

## 3 WITA: Our Partially-Aligned Dataset

We automatically harvest some partially-aligned data from Wikipedia and Wikidata and prepare a dataset called WITA. We select each Wikipedia article's first sentence from the 20190920 Wikipedia dump[3] as the target text. Then, we remove irrelevant tags and links with several predefined rules. We propose a KB extractor, as illustrated in Fig. 3, which can take the selected Wikipedia sentences and extract the corresponding KB triples. In the KB extractor, named entities are detected by an entity detector. The detected named entities are then combined into pairs by the Cartesian Product operation. The triples that mention these entity pairs are retrieved by a triple retriever that searches the corresponding KB triples from the Wikidata

[3]https://dumps.wikimedia.org/enwiki/

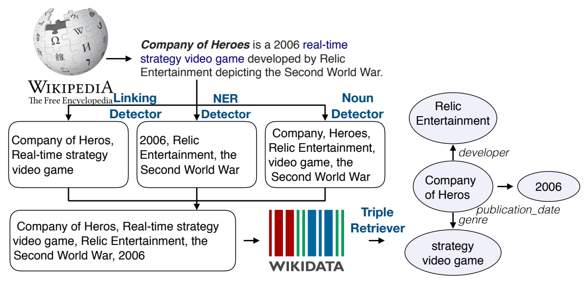
*Company of Heroes* is a 2006 real-time strategy video game developed by Relic Entertainment depicting the Second World War.
WIKIPEDIA The Free Encyclopedia
Linking Detector
NER Detector
Noun Detector
Company of Heros, Real-time strategy video game
2006, Relic Entertainment, the Second World War
Company, Heroes, Relic Entertainment, video game, the Second World War
Company of Heros, Real-time strategy video game, Relic Entertainment, the Second World War, 2006
WIKIDATA
Triple Retriever
Relic Entertainment
developer
Company of Heros
2006
publication_date
genre
strategy video game

Figure 3: Our proposed KB extractor for harvesting the partially-aligned data from Wikipedia and Wikidata.

database. We use an entity-recall based score to filter inappropriate sentences.

### 3.1 Entity Detector

We use three sub-detectors to recognize named entities and union them together. We first use a NER detector based on the spaCy[4]'s NER tool to recognize the named entities. Then we use the Noun detector based on spaCy's noun chunks recognition component to identify noun chunks. This detector is used because noun chunks have a high probability of being named entities. Finally, we use a linking detector, which is rule-based, to extract entities tagged with internal links. The detected entities for given sentence $c$ is denoted as $E_c = [e_1, e_2, \cdots, e_p]$, while $p$ is the entity number.

### 3.2 Triple Retriever

In order to quickly retrieve related triples for given named entities, we first store the Wikidata database in Elasticsearch[5]. We concatenate all possible variant names for an entity in Wikidata as the entity name. For example, *Steve Jobs*" has alternative names like "*Steven Paul Jobs*" and "*Steven Jobs*". The entity name is concatenated as "*Steve Jobs — Steven Paul Jobs — Steven Jobs*". In the Wikidata database, each triple contains a head, a relation and a tail entity and we denote the set of all the preprocessed triple as $\mathcal{D} = \{\langle h_i, r_i, t_i\rangle | \forall i\}$, where $h_i, r_i, t_i$ are the head, relation and tail entity for the $i$th triple.

For given detected entities $E_c$, we make a list of named entity pairs by conducting a Cartesian Product as $C_e = \{\langle e_i, e_j\rangle | \forall e_i \in E_c, e_j \in E_c, e_i \neq e_j\}$. Afterwards, we query the Wikidata database to find a triple that matches the given named entity pair $\langle e_i, e_j\rangle \in C_e$ to make the head entity close to $e_i$ while the tail entity close to $e_j$. It should be noted that the relation may be wrong, i.e. the matching triple describes a relation different from the one in the input sentence. But in reality, this probability is very small since most of the entity pairs only have one corresponding relation. For a given named entity pair $\langle e_i, e_j\rangle$, the query condition can be formally expressed as:

$$\begin{aligned}
\widehat{\langle h,r,t\rangle} = \arg\max_{\langle h,r,t\rangle} d(g(e_i,h)+g(e_j,t))+ \\
(1-d)(g(e_j,h)+g(e_i,t)) \\
\text{s.t. } (1-d)M + l(e_i,h) \geqslant \kappa;\ \ dM + l(e_i,t) \geqslant \kappa; \\
(1-d)M + l(e_j,t) \geqslant \kappa;\ \ dM + l(e_j,h) \geqslant \kappa; \\
\langle h,r,t\rangle \in \mathcal{D};\quad d \in \{0,1\},
\end{aligned}$$

in which $g$ is a single-term matching score[6] while $l$ is the string similarity metric ranging from 0 to 1. $M$ is a sufficiently large number and $d$ is an integer. $\kappa$ is a threshold preventing the retrieved head and tail being too different from $e_1$ and $e_2$. After we have retrieved entities for all sentences, we calculate a score based on entity-recall to filter wrongly extracted data-text pairs. entity-recall for KB triples and the corresponding text is defined as

$$r_e = \sum_{j=1}^{m} \mathbb{1}\{\sum_{i=1}^{n}[\mathbb{1}(w_j \in h_i) + \mathbb{1}(w_j \in r_i) + \mathbb{1}(w_j \in t_i)] > 0\}/m,$$

where $m$ is the length of the sentence while $n$ is the triple number. $r_e$ indicates how much information in text has been covered by the retrieved triples.

Since WebNLG is the most similar task to our PADTG task among others, we compare the statistics of our WITA dataset with WebNLG in Table 1. It can be observed that (1) WITA is larger than the WebNLG dataset making it more practical. It can be easily extended to more domains. (2) WITA contains more relation types and entity types than that of WebNLG, indicating that our dataset involves more domains. (3) The vocabulary of the target sentences of WITA is much larger than that of WebNLG, which shows that our dataset is more challenging and more realistic. (4) The entity-recall score of WITA is lower than WebNLG. This is because WITA is automatically annotated and some information in the text is not contained in the KB triples. The low entity-recall score causes the over-generation problem and the specific value measures how serious the problem is.

[4]https://spacy.io/
[5]https://www.elastic.co/products/elasticsearch
[6]https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html

|  | WITA | WebNLG |
|---|---|---|
| Size | 55,400 | 42,892 |
| Relation Type | 640 | 373 |
| Entity Type | 128,405 | 3,114 |
| Text Length | (18.8, 17.0, 5, 59) | (23.7, 22.0, 4, 84) |
| KB Number | (3.0, 3.0, 1, 11) | (2.9, 3.0, 1, 7) |
| Vocabulary | 102,404 | 8,886 |
| entity-recall | 0.508 | 0.625 |

Table 1: Statistics of WITA and WebNLG. For the text length and KB number, the data are mean, median, min and max respectively.

|  | BLEU | NIST | METEOR | ROUGE$_L$ | CIDEr |
|---|---|---|---|---|---|
| S2S | 0.463 | 7.97 | 0.385 | 0.693 | 4.12 |
| S2ST | 0.496 | 8.05 | 0.417 | 0.721 | 4.53 |
| DSG-A | 0.518 | 8.36 | 0.421 | 0.730 | 4.75 |
| DSG-H | 0.500 | 8.61 | 0.403 | 0.711 | 4.65 |
| DSG | **0.555** | **8.71** | **0.425** | **0.742** | **5.02** |
| DSG w/o RBS | 0.540 | 8.59 | 0.421 | 0.740 | 4.97 |
| DSG w/o SA | 0.522 | 8.38 | 0.421 | 0.734 | 4.83 |

Table 2: Main results.

## 4 Experiments

### 4.1 Experimental Setup

We split WITA into a training set, a development set, and a testing set of 50,000, 5,000, and 400 records respectively. For the purpose of evaluating the performance of the models, we ask human helpers to annotate the testing set sentences. The human helpers are asked to revise the input KB triples and the corresponding target sentences making them exactly consistent with each other. We use several evaluation metrics including BLEU (Papineni et al., 2002), ROUGE$_L$ (Lin, 2004), METEOR (Banerjee and Lavie, 2005), NIST (Doddington, 2002) and CIDEr (Vedantam et al., 2015) with the package provided by Novikova et al. (2017). We follow the default setting in ROUGE$_L$ where $\beta$ is set to 1.2. We build our model based on the Transformer model (Vaswani et al., 2017; Ott et al., 2019) and use Byte Pair Encoding (BPE) (Sennrich et al., 2016) to build the subword dictionary. We use Fairseq (Ott et al., 2019) to build our model and keep all hyper-parameters for Transformer unchanged. We set $\kappa = 0.75$ from $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ by extracting samples and ask human helper to evaluate. We use grid search to tune hyper-parameters on the development set and choose $\omega_w = 0.05$ from {0.02,0.05,0.1,0.2,0.5,1.0,2.0,5.0}, choose $\omega_c = 1.0$ from {0.02,0.05,0.1,0.2,0.5,1.0, 2.0,5.0} and choose $\alpha = 0.1$ from {0.02,0.05,0.1,0.2,0.5,1.0}. The model has 49M parameters and it takes 2.4 hours to train it on a NVIDIA TITAN RTX graphics card.

### 4.2 Comparison Models

We compare our full DSG model with the following baselines, state-of-the-art models, and ablations.

**S2S** utilizes the traditional S2S model (Sutskever et al., 2014; Cho et al., 2014) equipped with attention (Bahdanau et al., 2014; Luong et al., 2015) and copy (Gu et al., 2016) mechanism. It is recognised as the state-of-the-art model (Shimorina and Gardent, 2018) in the WebNLG (Gardent et al., 2017b) task.

**S2ST** utilizes the prevalent Transformer model (Vaswani et al., 2017; Ott et al., 2019) which outperforms the traditional S2S model in many generation tasks.

**DSG-A** utilizes the attention adaptor which adapts attention as the supportiveness scores in the loss.

**DSG-H** is almost the same as our DSG model. The only difference is that the supportiveness scores are adapted with hard adaptor while our DSG model uses the soft adaptor.

**DSG w/o RBS** is an ablation model. It removes the Rebalanced Beam Search component from our DSG model.

**DSG w/o SA** is an ablation model without the Supportiveness Adaptor. The supportiveness scores are only used by RBS in the generation phase. They are not adapted to the training loss.

### 4.3 Experimental Results

**Main Results.** The experimental results are shown in Table 2. It can be observed that our DSG model outperforms all comparison models in all metrics significantly and consistently, illustrating the effectiveness and consistentness of our framework. We can draw the following conclusions. (1) The superior performance of our DSG model shows that the supportiveness scores do help alleviate the over-generation problem. (2) The DSG-A model outperforms models without any adaptor. It fails to exceed the DSG model in all metrics. It shows attention can also be used to alleviate the over-generation problem but it performs not as good as our supportiveness scores. (3) The DSG model outperforms the DSG-H model illustrating that the soft adaptor is better than the hard adaptor. (4) The ablation experiments show that both the RBS and

| | 1-gram | 2-gram | 3-gram | 4-gram | 5-gram |
|---|---|---|---|---|---|
| S2ST | 962 | 2,313 | 3,118 | 3,425 | 3,501 |
| DSG-A | 894 | 2,161 | 2,934 | 3,217 | 3,290 |
| DSG-H | 646 | 1,817 | 2,494 | 2,786 | 2,854 |
| DSG | 741 | 1,894 | 2,599 | 2,870 | 2,936 |

Table 3: N-gram statistics for over-generation error analysis.

| | S2ST | | DSG | |
|---|---|---|---|---|
| | BLEU | ROUGE$_L$ | BLEU | ROUGE$_L$ |
| 10k | 0.215 | 0.467 | 0.260 | 0.514 |
| 20k | 0.383 | 0.631 | 0.421 | 0.659 |
| 30k | 0.446 | 0.684 | 0.487 | 0.711 |
| 40k | 0.500 | 0.719 | 0.518 | 0.728 |
| 50k | 0.496 | 0.721 | 0.555 | 0.749 |

Table 4: Dataset size analysis.

the SA components contribute to alleviating the over-generation problem.

Specifically, the S2S model performs worse than other models. It shows that the Transformer based model works better in this task. This observation is consistent with the results observed in a lot of other similar tasks. The S2ST model performs worse than all other transformer-based models. The reason is that it suffers severely from the over-generation problem and is very likely to generate superfluous content in the generation procedure. The DSG-A model outperforms other models without any adaptor. This is because attention can also be regarded as a kind of supportiveness and it can be undoubtedly used to detect the over-generated words. However, since the purpose of the attention is to give weights to each input word, it is forced to assign weights to input words even no input data support the target word. As a result, it performs worse than our DSG model. Our DSG model with a soft adaptor outperforms the DSG-H model equipped with a hard adaptor. The reason is that when the hard adaptor is used, some words are directly ignored possibly resulting in generating an incoherent target sentence. Therefore, though it outperforms other models without any supportiveness adaption, it fails to exceed our proposed DSG model. The ablation experiment results show that both the RBS and SA components contribute to alleviating the over-generation problem. SA mainly focuses on alleviating the problem in the training phase while RBS focuses on solving it in the generation phase. They are all essential components of our model.

**Supportiveness Distribution Analysis.** To give an intuitive understanding of how the supportiveness works and how it differs from the attention, we give an example comparing supportiveness distribution with the corresponding attention distribution. As shown in Fig. 4, it can be observed that both the supportiveness and the attention can capture the alignment relationship between words in the KB triples and words in the target sentence. The difference is that the aggregated supportiveness is high for correct words while the aggregated attention is not that significant for almost all words. The reason is that attention focuses on assigning weights to individual words with the weights summing to 1 while our proposed supportiveness score just focuses on deciding whether one word is correct or not. As a result, if one word is supported by many words in KB triples, our proposed supportiveness will be very high, while the attention may be relatively lower since its sum is fixed to 1. For example, three words "*Goose*" in the source sequence support the same word in the target text. The supportiveness for the word is very high while the attention is low since the three words in the source sequence dissipate the attention weight.

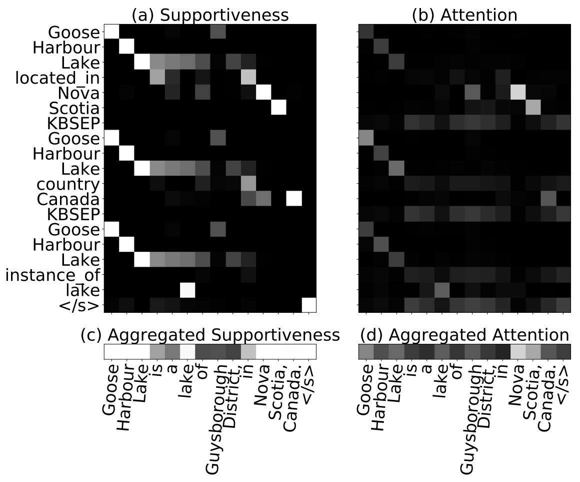

Figure 4: Comparision for supportiveness and attention. x-axis is the target text while y-axis is the given input KB triples. White stands for high score while black stands for low score. (a) and (b) show how each word in KB triples and text is aligned. (c) and (d) show the aggregated supportiveness and attention for each word in the target text.

**Over-Generation Error Analysis.** In order to perform a quantitative analysis of the over-generation problem and show how it is mitigated by our model, we conduct an over-generation error analysis in which we count all over-generated n-gram words to measure how serious the problem

| KB Triple | S2ST | DSG-H | DSG | Gold |
|---|---|---|---|---|
| ⟨Four Crowned Martyrs, genre, sculptural group⟩, ⟨Nanni di Banco notable_work, Four Crowned Martyrs⟩ | The Four Crowned Martyrs ( also known as the Four Crowned Martyrs ) is a sculptural group four by Nanni di Banco. | "Four Crowned Martyrs" a sculptural group Nanni di Banco. | Four Crowned Martyrs is a sculptural group by Nanni di Banco. | Four Crowned Martyrs is a sculptural group by Nanni di Banco. |
| ⟨Newfoundland and Labrador Route 341, located_in, Newfoundland and Labrador⟩ | Route 341 is a rural road in the Canadian province of Newfoundland and Labrador. | Route 341 a Canadian of Newfoundland and Labrador. | Route 341 is a settlement in Newfoundland and Labrador. | Route 341 is located in in Newfoundland and Labrador. |
| ⟨Gaius Helen Mohiam, creator, Frank Herbert⟩, ⟨Gaius Helen Mohiam, instance_of, fictional character⟩, ⟨Dune universe, creator, Frank Herbert⟩, ⟨Dune universe, instance_of, fictional universe⟩, ⟨ Gaius Helen Mohiam, from_fictional_universe, Dune universe⟩ | Gaius Helen Mohiam is a fictional character appearing in American comic books published by Frank Herbert. | Gaius Helen Mohiam is a fictional character created by Frank Herberfor the Dune univer. | Gaius Helen Mohiam is a fictional character in the Dune universe stationed by Frank Herbert. | Gaius Helen Mohiam is a fictional character in the Dune universe created by Frank Herbert. |

Table 5: Case study. The red font stands for over-generated words while the blue underline indicates incoherent parts.

is. For generated sentences, we first remove all stopwords and check whether each of the remaining words appears in the given input KB triple. If it is not contained in the KB triple, we will count it as an over-generated word. The statistics are shown in Table 3. It can be observed that the DSG-H has minimal over-generated words. This is because it directly drops all the possible over-generated words in the training. It gets the lowest over-generated words count at the cost of making the result less human-readable and thus has lower scores in other metrics like BLEU, etc. Our proposed DSG model outperforms all other models without significantly losing readability indicating that our proposed framework directly helps alleviate the over-generation problem.

**Dataset Size Analysis.** In order to explore whether our framework is capable of working on small datasets, we conduct a dataset size analysis. The results are shown in Table 4. It can be concluded that as the data size increases, all the performance of models with or without supportiveness improve noticeably. It shows that increasing data size help improve the overall scores. On the other hand, models assembled with supportiveness scores always outperform models without it. It shows that our novel architecture alleviates the over-generation problem at all scales of data size.

**Human Evaluation.** We conduct a human evaluation to eval the generation performance. We sample 130 samples from each model's generated sentences and ask human helpers to give an overall score and a match score with respect to the target sentences ranging from 1 to 10. The results are illustrated in Table 6. It can be concluded from the experiment that the DSG model generates better sentences in the sense of humans.

| | Overall | Match |
|---|---|---|
| S2ST | 7.315 | 7.231 |
| DSG-H | 7.285 | 7.331 |
| DSG | 7.377 | 7.569 |

Table 6: Human evaluation.

**Case Study.** We provide a case study for several models. As shown in Table 5, The S2ST model is always generating text accompanied with over-generated content while our proposed DSG model alleviates this problem significantly and consistently. When comparing the DSG-H model with the DSG model, we can find that the DSG-H model can also avoid producing over-generated content. However, it tends to remove a lot of correct words making the sentence incoherent and unreadable. Take the last case for example: The S2ST model conveys that Gaius Helen Mohiam comes from an American comic book. However, the given input KB triple does not mention this fact. On the other hand, the DSG-H model produces " *... created by Frank Herberfor the Dune univer ...* " which is even not a human-readable sentence.

## 5 Related Works

During the past few years, many tasks have been proposed to generate human-readable text from the structured data. WebNLG (Gardent et al., 2017a,b; Ferreira et al., 2019) is proposed to describe KB triples sampled from DBPedia (Auer et al., 2007).

The E2E (Novikova et al., 2017; Dušek et al., 2020) task is proposed for generating restaurant reviews based on the given attributes. Lebret et al. (2016) propose the Wikibio task to generate people's biography based on given Wikipedia infobox. Fu et al. (2020a) propose to generate text based on event chains. Moreover, Liang et al. (2009) propose to generate weather reports for weather records and Wiseman et al. (2017), Chen and Mooney (2008) and Puduppully et al. (2019) propose to generate a match report according to the match briefing. All these datasets are restricted to a few domains where well-aligned data is happened to be available. No existing works are focusing on handling partially-aligned data. To solve the dataset scarcity problem, Fu et al. (2020c) propose to use dual learning to train generation models based on unaligned text and knowledge triples. The model generates text based on input triples and then predict the input triples with a dual extraction model. The two models are trained alternatively with dual learning. Although Cheng et al. (2020) proposed the ENT-DESC task aiming at generating better text description for a few entities by exploring the knowledge from KB, their focus is more on distilling the useful part from the input knowledge.

Text aligning has been studied for many years. Dyer et al. (2013) propose the Fast Align model which is a log-linear reparameterization of IBM Model 2. Legrand et al. (2016) propose a new score aggregation method to improve the alignment result. Moreover, attention-based models (Bahdanau et al., 2014) can also be recognized as a kind of alignment. However, these models focus on aligning source words to target words, and no existing models have been proposed to directly calculate supportiveness for generation tasks. In generation systems, Fu et al. (2020b) propose to dynamically align the current generation state with topics to improve the generation performance. However, it still can not directly align to the input source words.

## 6 Conclusions

In this work, we propose a new task, namely, partially-aligned Data-to-Text generation, in which we generate human-readable text based on automatically produced training data. This task is more practical and extensible to any domains. We propose a distant supervision generation framework that tackling the task. The experimental results show that our proposed model solves the over-generation problem effectively and outperforms all baseline models. Moreover, we contribute a partially-aligned dataset WITA produced by our novel automatically annotating framework which is suitable for this new task.

## Acknowledgments

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) and the National Natural Science Foundation of China (NSFC No. 61532010, 61732008).

## 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.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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.

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.

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

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.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 552–562.

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, Lidong Bing, and Wai Lam. 2020c. Unsupervised kb-to-text generation with auxiliary triple extraction using dual learning. In *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*.

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.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

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*, pages 1203–1213.

Joël Legrand, Michael Auli, and Ronan Collobert. 2016. Neural network-based word alignment through score aggregation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 66–73.

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.

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

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

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.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

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.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Anastasia Shimorina and Claire Gardent. 2018. Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370.

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.

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*, pages 2253–2263.