ENT-DESC: Entity Description Generation by Exploring Knowledge Graph

Liying Cheng[*1,2], **Dekun Wu**[†3], **Lidong Bing**[2], **Yan Zhang**[†1], **Zhanming Jie**[†1], **Wei Lu**[1], **Luo Si**[2]
[1] Singapore University of Technology and Design
[2] DAMO Academy, Alibaba Group [3] York University, Canada
{liying.cheng, l.bing, luo.si}@alibaba-inc.com, jackwu@eecs.yorku.ca,
{yan_zhang, zhanming_jie}@mymail.sutd.edu.sg, luwei@sutd.edu.sg

Abstract

Previous works on knowledge-to-text generation take as input a few RDF triples or key-value pairs conveying the knowledge of some entities to generate a natural language description. Existing datasets, such as WIKIBIO, WebNLG, and E2E, basically have a good alignment between an input triple/pair set and its output text. However, in practice, the input knowledge could be more than enough, since the output description may only cover the most significant knowledge. In this paper, we introduce a large-scale and challenging dataset to facilitate the study of such a practical scenario in KG-to-text. Our dataset involves retrieving abundant knowledge of various types of main entities from a large knowledge graph (KG), which makes the current graph-to-sequence models severely suffer from the problems of information loss and parameter explosion while generating the descriptions. We address these challenges by proposing a multi-graph structure that is able to represent the original graph information more comprehensively. Furthermore, we also incorporate aggregation methods that learn to extract the rich graph information. Extensive experiments demonstrate the effectiveness of our model architecture. [1]

1 Introduction

KG-to-text generation, automatically converting knowledge into comprehensive natural language, is an important task in natural language processing (NLP) and user interaction studies (Damljanovic et al., 2010). Specifically, the task takes as input some structured knowledge, such as resource description framework (RDF) triples of

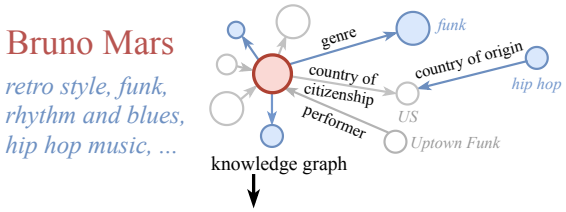


Figure 1: An example showing our proposed task.

WebNLG (Gardent et al., 2017), key-value pairs of WIKIBIO (Lebret et al., 2016) and E2E (Novikova et al., 2017), to generate natural text describing the input knowledge. In essence, the task can be formulated as follows: given a main entity, its one-hop attributes/relations (e.g., WIKIBIO and E2E), and/or multi-hop relations (e.g., WebNLG), the goal is to generate a text description of the main entity describing its attributes and relations. Note that these existing datasets basically have a good alignment between an input knowledge set and its output text. Obtaining such data with good alignment could be a laborious and expensive annotation process. More importantly, in practice, the knowledge regarding the main entity could be more than enough, and the description may only cover the most significant knowledge. Thereby, the generation model should have such differentiation capability.

In this paper, we tackle an entity description generation task by exploring KG in order to work towards more practical problems. Specifically, the aim is to generate a description with one or more sentences for a main entity and a few topic-related entities, which is empowered by the knowledge from a KG for a more natural description. In order to facilitate the study, we introduce a new dataset, namely *entity-to-description* (ENT-DESC)

[*] Liying Cheng is under the Joint Ph.D. Program between Alibaba and Singapore University of Technology and Design.

[†] Dekun Wu was a visiting student at SUTD. Yan Zhang and Zhanming Jie were interns at Alibaba.

[1] Our code and data are available at https://github.com/LiyingCheng95/EntityDescriptionGeneration.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1187–1197,
November 16–20, 2020.

extracted from Wikipedia and Wikidata, which contains over 110k instances. Each sample is a triplet, containing a set of entities, the explored knowledge from a KG, and the description. Figure 1 shows an example to generate the description of the main entity, i.e., *Bruno Mars*, given some relevant keywords, i.e., *retro style, funk,* etc., which are called topic-related entities of *Bruno Mars*. We intend to generate the short paragraph below to describe the main entity in compliance with the topic revealed by topic-related entities. For generating accurate descriptions, one challenge is to extract the underlying relations between the main entity and keywords, as well as the peripheral information of the main entity. In our dataset, we use such knowledge revealed in a KG, i.e., the upper right in Figure 1 with partially labeled triples. Therefore, to some extent, our dataset is a generalization of existing KG-to-text datasets. The knowledge, in the form of triples, regarding the main entity and topic entities is automatically extracted from a KG, and such knowledge could be more than enough and not necessarily useful for generating the output.

Our dataset is not only more practical but also more challenging due to lack of explicit alignment between the input and the output. Therefore, some knowledge is useful for generation, while others might be noise. In such a case that many different relations from the KG are involved, standard graph-to-sequence models suffer from the problem of low training speed and parameter explosion, as edges are encoded in the form of parameters. Previous work deals with this problem by transforming the original graphs into Levi graphs (Beck et al., 2018). However, Levi graph transformation only explicitly represents the relations between an original node and its neighbor edges, while the relations between two original nodes are learned implicitly through graph convolutional networks (GCN). Therefore, more GCN layers are required to capture such information (Marcheggiani and Perez-Beltrachini, 2018). As more GCN layers are being stacked, it suffers from information loss from KG (Abu-El-Haija et al., 2018). In order to address these limitations, we present a multi-graph convolutional networks (MGCN) architecture by introducing multi-graph transformation incorporated with an aggregation layer. Multi-graph transformation is able to represent the original graph information more accurately, while the aggregation layer learns to extract useful information from the KG. Extensive experiments are conducted on both our dataset and benchmark dataset (i.e., WebNLG). MGCN outperforms several strong baselines, which demonstrates the effectiveness of our techniques, especially when using fewer GCN layers.

Our main contributions include:

- We construct a large-scale dataset ENT-DESC for a more practical task of entity description generation by exploring KG. To the best of our knowledge, ENT-DESC is the largest dataset of KG-to-text generation.
- We propose a multi-graph structure transformation approach that explicitly expresses a more comprehensive and more accurate graph information, in order to overcome limitations associated with Levi graphs.
- Experiments and analysis on our new dataset show that our proposed MGCN model incorporated with aggregation methods outperforms strong baselines by effectively capturing and aggregating multi-graph information.

2 Related Work

Dataset and Task. There is an increasing number of new datasets and tasks being proposed in recent years as more attention has been paid to data-to-text generation. Gardent et al. (2017) introduced the WebNLG challenge, which aimed to generate text from a small set of RDF knowledge triples (no more than 7) that are well-aligned with the text. To avoid the high cost of preparing such well-aligned data, researchers also studied how to leverage automatically obtained partially-aligned data in which some portion of the output text cannot be generated from the input triples (Fu et al., 2020b). Koncel-Kedziorski et al. (2019) introduced AGENDA dataset, which aimed to generate paper abstract from a title and a small KG built by information extraction system on the abstracts and has at most 7 relations. In our work, we directly create a knowledge graph for the main entities and topic-related entities from Wikidata without looking at the relations in our output. Scale-wise, our dataset consists of 110k instances while AGENDA is 40k. Lebret et al. (2016) introduced WIKIBIO dataset that generates the first sentence of biographical articles from the key-value pairs extracted from the article's infobox. Novikova et al. (2017) introduced E2E dataset in the restaurant domain, which aimed to generate restaurant recommendations given 3 to 8 slot-value pairs. These two datasets were only

for a single domain, while ours focuses on multiple domains of over 100 categories, including people, event, location, organization, etc. Another difference is that we intend to generate the first paragraph of each Wikipedia article from a more complicated KG, but not key-value pairs. Another popular task is AMR-to-text generation (Konstas et al., 2017). The structure of AMR graphs is rooted and denser, which is quite different from the KG-to-text task. Researchers also studied how to generate texts from a few given entities or prompts (Li et al., 2019; Fu et al., 2020a). However, they did not explore the knowledge from a KG.

Graph-to-sequence Modeling. In recent years, graph convolutional networks (GCN) have been applied to several tasks (e.g., semi-supervised node classification (Kipf and Welling, 2017), semantic role labeling (Marcheggiani and Titov, 2017) and neural machine translation (Bastings et al., 2017)) and also achieved state-of-the-art performance on graph-to-sequence modeling. In order to capture more graphical information, Velickovic et al. (2017) introduced graph attention networks (GATs) through stacking a graph attentional layer, but only allowed to learn information from adjacent nodes implicitly without considering a more global contextualization. Marcheggiani and Titov (2017) then used GCN as the encoder in order to capture more distant information in graphs. Since there are usually a large amount of labels for edges in KG, such graph-to-sequence models without graph transformation will incur information loss and parameter explosion. Beck et al. (2018) proposed to transform the graph into Levi graph in order to work towards the aforementioned deficiencies, together with gated graph neural network (GGNN) to build graph representation for AMR-to-text problem. However, they face some new limitations brought in by Levi graph transformation: the entity-to-entity information is being ignored in Levi transformation, as also mentioned in their paper. Afterwards, deeper GCNs were stacked (Guo et al., 2019) to capture such ignored information implicitly. In contrast, we intend to use fewer GCN layers to capture more global contextualization by explicitly stating all types of graph information with different transformations.

3 Task Description

In this paper, we tackle a practical problem of entity description generation by exploring KG. In practice, it is difficult to describe an entity in only a few sentences as there are too many aspects for an entity. Now, if we are given a few topic-related entities as topic restrictions to the main entity, the text to be generated could be more concrete, particularly when we are allowed to explore the connections among these entities in KG. As seen in Figure 1, when we are asked to use one or two sentences to introduce "*Bruno Mars*"[2], his popular singles will first come into some people's minds, while his music genres might be in other people's first thought. With the introduction of topic-related entities, the description will have some focus. In this case, when topic-related entities, i.e., *R&B*, *hip hop*, *rock*, etc., are provided, we are aware of describing *Bruno Mars* in the direction of music styles on top of their basic information.

Formally, given a set of entities $\mathbf{e} = \{\mathrm{E}_1, ..., \mathrm{E}_n\}$ and a KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where E_1 is main entity, $\mathrm{E}_2, ..., \mathrm{E}_n$ are topic-related entities, $\mathcal{V}$ is the set of entity nodes and $\mathcal{E}$ is the set of directed relation edges. We intend to generate a natural language text $\mathbf{y} = \{y_1, y_2, \cdots, y_T\}$. Meanwhile, we explore $\mathcal{G}$ for useful information to allow a more natural description. Here, the KG $\mathcal{G}$ can also be written as a set of RDF triples: $\mathcal{G} = \{\langle \mathrm{V}_{\mathrm{S}_1}, \mathrm{P}_1, \mathrm{V}_{\mathrm{O}_1} \rangle, ..., \langle \mathrm{V}_{\mathrm{S}_\mathrm{M}}, \mathrm{P}_\mathrm{M}, \mathrm{V}_{\mathrm{O}_\mathrm{M}} \rangle\}$, where M is the total number of triples, $\mathrm{V}_{\mathrm{S}_i}, \mathrm{V}_{\mathrm{O}_i} \in \mathcal{V}$ are the subject and object entities respectively, P_i is the predicate stating the relation between $\mathrm{V}_{\mathrm{S}_i}$ and $\mathrm{V}_{\mathrm{O}_i}$.

	WebNLG	AGENDA	E2E	ENT-DESC
# instances	43K	41K	51K	110K
Input vocab	4.4K	54K	120	420K
Output vocab	7.8K	78K	5.2K	248K
# distinct entities	3.1K	297K	77	691K
# distinct relations	358	7	8	957
Avg. # triples per input	3.0	4.4	5.6	27.4
Avg. # words per output	23.7	141.3	20.3	31.0

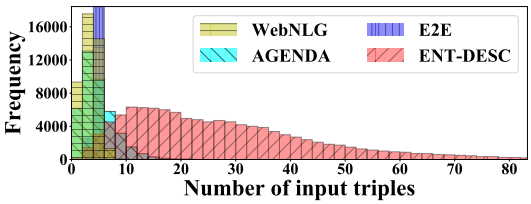
Table 1: Dataset statistics of WebNLG, AGENDA and our prepared ENT-DESC.

4 ENT-DESC Dataset

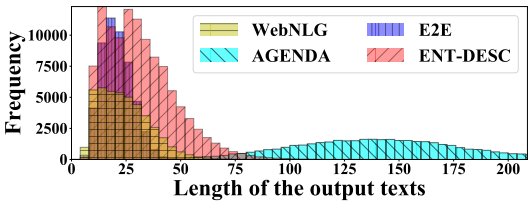
To prepare our dataset, we first use Nayuki's implementation[3] to calculate the PageRank score for more than 9.9 million Wikipedia pages. We then extract the categories from Wikidata for the top 100k highest scored pages and manually select 90 categories out of the top 200 most frequent ones as

[2] https://en.wikipedia.org/wiki/Bruno_Mars

[3] https://www.nayuki.io/page/computing-wikipedias-internal-pageranks



(a) Number of the input triples in each dataset.



(b) Length of the output texts in each dataset.

Figure 2: Dataset comparison among WebNLG, AGENDA, E2E and our ENT-DESC.

the seed categories. The domains of the categories mainly include humans, events, locations and organizations. The entities from these categories are collected as our candidate set of main entities. We further process their associated Wikipedia pages for collecting the first paragraphs and entities with hyperlink as topic-related entities. We then search Wikidata to gather neighbors of the main entities and 1-hop/2-hop paths between main entities and their associated topic-related entities, which finally results in a dataset consisting of more than 110k entity-text pairs with 3 million triples in the KG. Although more-hop paths might be helpful, we limit to 1-hop/2-hop paths for the first study. The comparison of our dataset with WebNLG, AGENDA and E2E is shown in Table 1 and Figure 2.

In the comparison of these four datasets, there are some obvious differences. First, our dataset is significantly larger than WebNLG, AGENDA and E2E (i.e., more than twice of their instances). Meanwhile, our vocabulary size and numbers of distinct entities/relations are all much larger. Second, the average number of input triples per instance is much larger than those of the other two. More importantly, our dataset provides a new genre of data for the task. Specifically, WebNLG has a strict alignment between input triples and output text, and accordingly, each input triple roughly corresponds to 8 words. AGENDA is different from WebNLG for generating much longer output, namely paper abstracts, with the paper title also given as input. Moreover, as observed, quite a portion of text information cannot be directly covered by the input triples. E2E focuses on the restaurant domain with relatively simple inputs, including 77 entities and 8 relations in total. Considering the construction details of these 3 datasets, all their input triples provide useful information (i.e., should be used) for generating the output. In contrast, our dataset has a much larger number of input triples, particularly considering the length difference of output texts. Lastly, another unique characteristic of our dataset is that not every input triple is useful for generation, which brings in the challenge that a model should be able to distill the helpful part for generating a better output sequence.

5 Our MGCN Model

Given the explored knowledge, our task can be cast as a problem of generating text from KG. We propose an encoder-decoder architecture with a multi-graph transformation, shown in Figure 3.

5.1 Multi-Graph Encoder

We first briefly introduce the general flow of multi-graph encoder which consists of n MGCN layers. Before the first layer, graph embedding $\mathbf{h}^{(0)}$ representing a collection of node embeddings is initialized from input KG after multi-graph transformation. By stacking n MGCN layers accordingly with multi-graph transformation and aggregation, we obtain the final graph representation by aggregating the outputs of n MGCN layers for decoding. We explain the details of an MGCN layer as follows.

Graph Encoder. Before introducing our multi-graph transformation, we first look at our basic graph encoder in each MGCN layer (i.e., Graph Encoder 1 to 6 in Figure 3 left). In this paper, we adopt graph convolutional networks (GCNs) (Duvenaud et al., 2015; Kearnes et al., 2016; Kipf and Welling, 2017; Marcheggiani and Titov, 2017) as the basic encoder to consider the graph structure and to capture graph information for each node. More formally, given a directed graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$, we define a feature vector $\mathbf{x}_{\mathrm{v}} \in \mathbb{R}^d$ for each node $\mathrm{v} \in \mathcal{V}^*$. In order to capture the information of neighbors $\mathcal{N}(\cdot)$, the node representation $\mathbf{h}_{\mathrm{v}_j}$ for each $\mathrm{v}_j \in \mathcal{V}^*$ is calculated as:

$$\mathbf{h}_{\mathrm{v}_j} = ReLU\Big(\sum_{\mathrm{v}_i \in \mathcal{N}(\mathrm{v}_j)} W_{\mathrm{p}(i,j)}\mathbf{x}_{\mathrm{v}_i} + \mathbf{b}_{\mathrm{p}(i,j)}\Big),$$

where $\mathrm{p}(i,j)$ denotes the edge between node v_i and v_j including three possible directions: (1) v_i to v_j, (2) v_j to v_i, (3) v_i to itself when i equals to

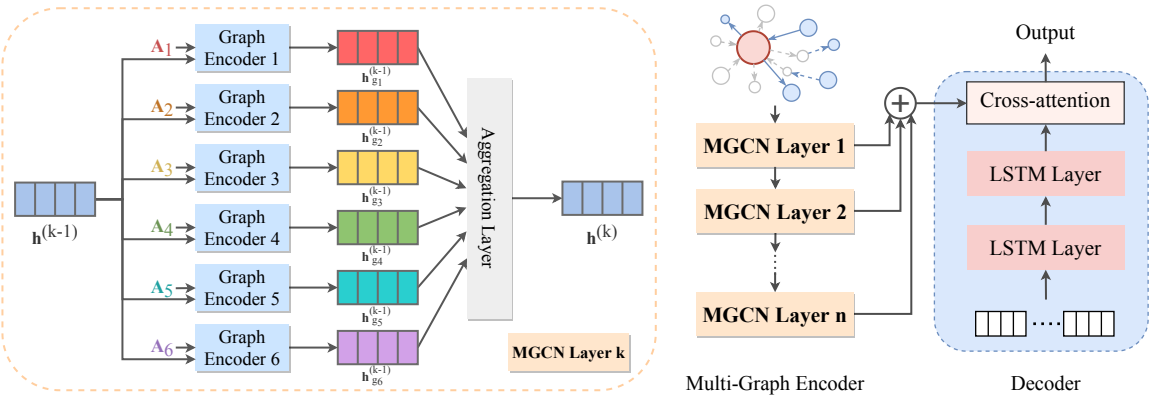


Figure 3: Overview of our model architecture. There are n MGCN layers in the multi-graph encoder, and 2 LSTM layers in the decoder. $\mathbf{h}^{(k-1)}$ is the input graph representation at Layer k, and its 6 copies together with the corresponding adjacent matrices $\mathbf{A}_i$'s of transformed graphs in the multi graph (refer to Figure 4) are fed into individual basic encoders. Finally, we obtain the graph representation $\mathbf{h}^{(k)}$ for the next layer by aggregating the representations from these encoders.

j. Weight matrix $W \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b} \in \mathbb{R}^d$ are model parameters. $ReLU$ is the rectifier linear unit function. Only immediate neighbors of each node are involved in the equation above as it represents a single-layer GCN.

Multi-Graph Transformation. The basic graph encoder with GCN architecture as described above struggles with the problem of parameter explosion and information loss, as the edges are encoded in the form of parameters. Previous works (Beck et al., 2018; Guo et al., 2019; Koncel-Kedziorski et al., 2019) deal with this deficiency by transforming the graph into a Levi graph. However, Levi graph transformation also has its limitations, where entity-to-entity information is learned implicitly. In order to overcome all the difficulties, we introduce a multi-graph structure transformation. A simple example is shown in Figure 4. Given such a directed graph, where $\mathrm{E}_1, \mathrm{E}_2, \mathrm{E}_3, \mathrm{E}_4$ represent entities and $\mathrm{R}_1, \mathrm{R}_2, \mathrm{R}_3$ represent relations in the KG, we intend to transform it into multiple graphs which capture different types of information. Similar to Levi graph transformation, all the entities and relations are represented as nodes in our multi-graph structure. By doing such transformation, we are able to represent relations in the same format as entities using embeddings directly, which avoids the risk of parameter explosion. This multi-graph transformation can be generalised for any graph regardless of the complexity and characteristic of the KG, and the transformed graph can be applied to any model architecture.

In this work, we employ a six-graph structure for our multi-graph transformation as shown in Figure 4. Firstly, in self graph (1), each node is assigned a self-loop edge namely *self* label. Secondly, graphs (2) and (3) are formed by connecting the nodes representing the entities and their adjacent relations. In addition to connecting them in their original direction using *default1* label, we also add a *reverse1* label for the inverse direction of their original relations. Thirdly, we create graphs (4) and (5) by connecting the nodes representing adjacent entities in the input graph, labeled by *default2* and *reverse2*, respectively. These two graphs overcome the deficiency of Levi graph transformation by explicitly representing the entity-to-entity information from the input graph. It also allows us to differentiate entities and relations by adding edges between entities. Finally, in order to consider more global contextualization, we add a global node on top of the graph structure to form graph (6). Each node is assigned with a *global* edge directed from global node. In the end, the set of transformed graphs can be represented by their edge labels $\mathcal{T} = \{self, default, reverse, default2, reverse2, global\}$.

Given the six transformed graphs mentioned above, we construct six corresponding adjacency matrices: $\{\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_6\}$. As shown in Figure 3 (left), these adjacency matrices are used by six basic graph encoders to obtain the corresponding transformed graph representations (i.e., $\mathbf{h}_g$).

Aggregation Layer. After learning 6 embeddings of multi graphs from the basic encoders at the current MGCN layer $k-1$, the model goes through an aggregation layer to obtain the graph

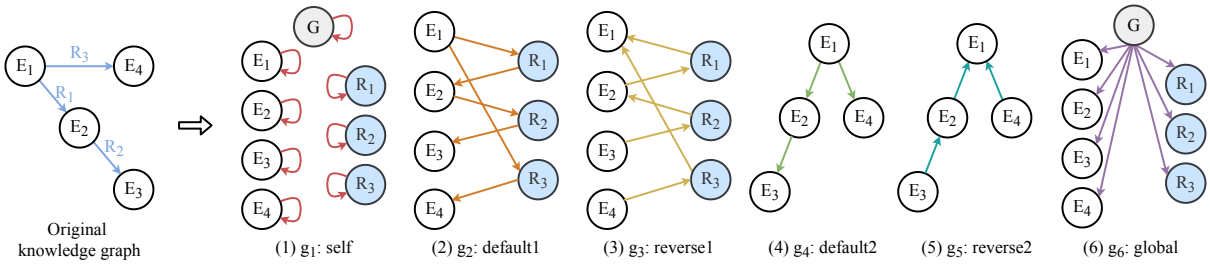


Figure 4: An example of multi-graph transformation.

embedding for the next MGCN layer k. We can get it by simply concatenating all 6 transformed graph embeddings with different types of edges. However, such simple concatenation of the transformed graphs involves too many features and parameters. In order to address the challenge mentioned above, we propose three aggregation methods for the multi-graph structure: sum-based, average-based and CNN-based aggregation.

Firstly, in sum-based aggregation layer, we compute the representation $\mathbf{h}^{(k)}$ at k-th layer as:

$$\mathbf{h}^{(k)} = \sum_{g_i \in \mathcal{T}} \mathbf{h}_{g_i}^{(k-1)},$$

where $\mathbf{h}_{g_i}^{(k-1)}$ represents the i-th graph representation, and $\mathcal{T}$ is the set of all transformed graphs. Sum-based aggregation allows a linear approximation of spectral graph convolutions and helps to reduce data sparsity and over-fitting problems.

Similarly, we apply an average-based aggregation method by normalizing each graph through a mean operation:

$$\mathbf{h}^{(k)} = \frac{1}{m} \sum_{g_i \in \mathcal{T}} \mathbf{h}_{g_i}^{(k-1)},$$

where m is the number of graphs in $\mathcal{T}$.

We also try to employ a more complex CNN-based aggregation method. Formally, the representation $\mathbf{h}^{(k)}$ at k-th layer is defined as:

$$\mathbf{h}^{(k)} = W_{conv} \mathbf{h}_{mg}^{(k-1)} + \mathbf{b}_{mg}^{(k)} .$$

Here, we use convolutional neural networks (CNN) to convolute the multi-graph representation, where $\mathbf{h}_{mg} = [\mathbf{h}_{g_1}, ..., \mathbf{h}_{g_6}]$ is the representation of multi-graph and $\mathbf{b}_{mg}^{(k)}$ is the bias term.

By applying these aggregation methods, we obtain the graph representation for the next layer $\mathbf{h}^{(k)}$, which is able to capture different aspects of graph information more effectively by learning different types of edges in each transformed graph.

Stacking MGCN Layers. With the introduction of MGCN layer as described above, we can capture the information of higher-degree neighbors by stacking multiple MGCN layers. Inspired by Xu et al. (2018), we employ a concatenation operation over $\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(n)}$ to aggregate the graph representations from all MGCN layers (Figure 3 right) to form the final layer $\mathbf{h}^{(final)}$, which can be written as follows:

$$\mathbf{h}^{(final)} = \left[\mathbf{h}^{(1)}, \cdots \mathbf{h}^{(n)}\right].$$

Such a mechanism allows weight sharing across graph nodes, which helps to reduce overfitting problems. To further reduce the number of parameters and overfitting problems, we apply the softmax weight tying technique (Press and Wolf, 2017) by tying source embeddings and target embeddings with a target softmax weight matrix.

5.2 Attention-based LSTM Decoder

We adopt the commonly-used standard attention-based LSTM as our decoder, where each next word y_t is generated by conditioning on the final graph representation $\mathbf{h}^{(final)}$ and all words that have been predicted $y_1, ..., y_{t-1}$. The training objective is to minimize the negative conditional log-likelihood. Thus, the objective function can be written as:

$$\mathcal{L} = -\sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(y_t | y_1, ..., y_{t-1}, \mathbf{h}^{(final)}),$$

where T represents the length of the output sequence, and p is the probability of decoding each word y_t parameterized by $\boldsymbol{\theta}$. As shown in the decoder from Figure 3, we stack 2 LSTM layers and apply a cross-attention mechanism in our decoder.

6 Experiments

6.1 Experimental Settings

We implement our MGCN architecture based on MXNET (Chen et al., 2015) and Sockeye toolkit. Hidden units and embedding dimensions for both encoder and decoder are fixed at 360. We use Adam (Kingma and Ba, 2014) with an initial learning rate of 0.0003 and update parameters with a batch size of 16. The training phase is stopped when detecting the convergence of perplexity on the validation set.

Models	BLEU	METEOR	TER↓	$ROUGE_1$	$ROUGE_2$	$ROUGE_L$	PARENT
S2S (Bahdanau et al., 2014)	6.8	10.8	80.9	38.1	21.5	40.7	10.0
GraphTransformer (Koncel-Kedziorski et al., 2019)	19.1	16.1	94.5	53.7	37.6	54.3	21.4
GRN (Beck et al., 2018)	24.4	18.9	70.8	54.1	38.3	55.5	21.3
GCN (Marcheggiani and Perez-Beltrachini, 2018)	24.8	19.3	70.4	54.9	39.1	56.2	21.8
DeepGCN (Guo et al., 2019)	24.9	19.3	70.2	55.0	39.3	56.2	21.8
MGCN	**25.7**	**19.8**	**69.3**	**55.8**	**40.0**	**57.0**	**23.5**
MGCN + CNN	**26.4**	**20.4**	69.4	56.4	40.5	**57.4**	**24.2**
MGCN + AVG	26.1	20.2	**69.2**	56.4	40.3	57.3	23.9
MGCN + SUM	**26.4**	20.3	69.8	56.4	**40.6**	**57.4**	23.9
GCN + delex	28.4	22.9	65.9	61.8	45.5	62.1	30.2
MGCN + CNN + delex	29.6	**23.7**	**63.2**	**63.0**	**46.7**	**63.2**	**31.9**
MGCN + SUM + delex	**30.0**	**23.7**	67.4	62.6	46.3	62.7	31.5
The rows below are results of generating from entities only without exploring the KG.							
E2S	23.3	20.4	68.7	58.8	41.9	58.2	27.7
E2S + delex	21.8	20.5	67.5	59.5	39.5	59.2	23.4
E2S-MEF	24.2	21.3	65.8	59.8	43.3	60.0	26.3
E2S-MEF + delex	20.6	20.3	66.5	59.1	40.0	59.3	24.3

Table 2: Main results of models on ENT-DESC dataset. ↓ indicates lower is better.

During decoding, we use beam search with a beam size of 10. All models are run with V100 GPU.

We evaluate our models by applying both automatic and human evaluations. For automatic evaluation, we use several common evaluation metrics: BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), TER (Snover et al., 2006), $ROUGE_1$, $ROUGE_2$, $ROUGE_L$ (Lin, 2004), PARENT (Dhingra et al., 2019). We adapt MultEval (Clark et al., 2011) and Py-rouge for resampling and significance test.

6.2 Main Experimental Results

We present our main experiments on ENT-DESC dataset and compare our proposed MGCN models with various aggregation methods against several strong GNN baselines (Bahdanau et al., 2014), GraphTransformer (Koncel-Kedziorski et al., 2019), GRN (Beck et al., 2018), GCN (Marcheggiani and Perez-Beltrachini, 2018) and DeepGCN (Guo et al., 2019), as well as a sequence-to-sequence (S2S) baseline. We re-implement GRN, GCN and DeepGCN using MXNET. We rearrange the order of input triples following the occurrence of entities in output for S2S model to ease its limitation of not able to capture the graph structure. We also apply sequence-to-sequence models on generating outputs directly from entities without exploring KG by (1) randomly shuffling the order of all input entities (E2S) and (2) randomly shuffling the order of all topic-related entities while keeping the **M**ain **E**ntity at **F**ront (E2S-MEF). Furthermore, we apply a delexicalization technique on our dataset. We delexicalize the main entity and topic-related entities by replacing these entities with tokens indicating the entity types and indices.

Main results on our ENT-DESC dataset are shown in Table 2. Here, the numbers of layers in all baseline models and our MGCN models are set to be 6 for fair comparisons. Our models consistently outperform the baseline models on all evaluation metrics. S2S model has poor performance, mainly because the structure of our input triples is complicated as explained earlier. Compared to GRN and GCN models, the BLEU score of MGCN model increases by 1.3 and 0.9, respectively. This result suggests the effectiveness of multi-graph transformation, which is able to capture more comprehensive information compared to the Levi graph transformation used by GCN and GRN (especially entity-to-entity information in the original graph). We then apply multiple methods of aggregation on top of the multi-graph structure. MGCN+CNN and MGCN+SUM report the highest BLEU score of 26.4, followed by MGCN+AVG. By applying our delexicalization technique, the results are further boosted by 3.2 to 3.6 BLEU scores for both baseline and our proposed models. Moreover, our MGCN models and most baseline models outperform E2S and E2S-MEF, suggesting the importance of exploring KG when generating entity descriptions. Compared to E2S and E2S-MEF, there is no further improvement after applying delexicalization (i.e., E2S+delex and E2S-MEF+delex). We speculate it is because the copy mechanism is incorporated in the sequence-to-sequence model. Some useful information in original entities may be lost when further applying the delexicalization.

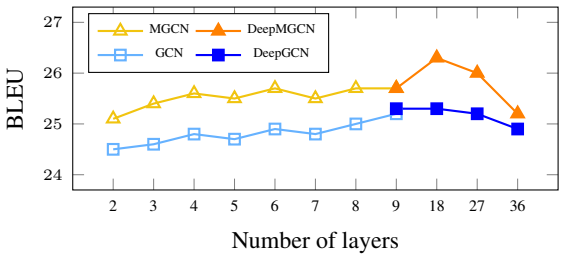


Figure 5: Effect of different numbers of layers.

# Input triples	# Instances	GCN	MGCN+SUM	Δ (BLEU)
1 to 10	1,790	19.4	21.3	+1.9
11 to 20	2,999	22.6	24.6	+2.0
21 to 30	2,249	23.2	25.0	+1.8
31 to 50	2,830	31.6	32.8	+1.2
51 to 100	1,213	23.9	24.7	+0.8

Table 3: Effect of different numbers of input triples.

Model	BLEU	Δ (BLEU)
MGCN + SUM	26.4	-
– g_6: *global*	26.0	-0.4
– g_5: *reverse2*	25.8	-0.6
– g_4: *default2*	26.1	-0.3
– g_3: *reverse1*	25.7	-0.7
– g_2: *default1*	26.1	-0.3
MGCN	25.7	-0.7
GCN	24.8	-1.4

Table 4: Results of the ablation study.

6.3 Analysis and Discussion

Effect of different numbers of MGCN layers. In order to examine the robustness of our MGCN models, we conduct further experiments by using different numbers of MGCN layers. The results are shown in Figure 5. We use MGCN to compare with the strongest baseline models using GCN according to the results in Table 2. More specifically, we compare to GCN on 2 to 9 layers and DeepGCN on 9, 18, 27 and 36 layers. As shown in Figure 5, both models perform better initially as more GCN/MGCN layers are being stacked and start to drop afterward. In general, MGCN/DeepMGCN achieves decent performance improvements of 0.3 to 1.0 from 2 to 36 layers, as shown in the line chart. DeepMGCN achieves 26.3 BLEU score at 18 MGCN layers, which is 1.0 higher than deepGCN. It shows that, compared with learning the information implicitly by Levi graph, our multi-graph transformation brings in robust improvements by explicitly representing all types of information in the graph. Another observation is that the BLEU score of MGCN with 3 layers (25.4) is already higher than the best performance of GCN/deepGCN.

Effect of various numbers of input triples. In order to have a deeper understanding of how multi-graph transformation helps the generation, we further explore the model performance under different numbers of triples on the test set. Table 3 shows the BLEU comparison between MGCN+SUM and GCN when using 6 layers. Both models perform the best when the number of triples is between 31 and 50. They both have a poorer performance when the number of triples is too small or too large, which should be due to the fact that the models have insufficient or very noisy input information for generation. Another observation is that the improvement of BLEU (Δ) by our model is greater with a smaller number of input triples. It is plausibly because when the graph is larger, although our transformation techniques still bring in overall BLEU improvements, the increased graph complexity due to the transformation also hinders the generation.

Ablation Study. To examine the impact of each graph in our multi-graph structure, we show the ablation study in Table 4. Each transformed graph is removed respectively from MGCN+SUM with 6 layers, except for the g_1 (*self*), which is always enforced in the graph (Kipf and Welling, 2017). We notice that the result drops after removing any transformed graph from the multi-graph. Particularly, we observe the importance of $\{default2, reverse2\}$ and $\{default1, reverse1\}$ are equivalent, as the BLEU scores after removing them individually are almost the same. This explains how multi-graph structure addresses the deficiency of Levi graph, i.e., entity-to-entity information is not represented explicitly in Levi graph. Additionally from the results, it is beneficial to represent the edges in the reverse direction for more effective information extraction in directed graphs as there are relatively larger gaps in BLEU drop after removing g_3 (*reverse1*) or g_5 (*reverse2*).

Case Study. Table 5 shows example outputs generated by GCN and MGCN+SUM, as compared to the gold reference. The main entity is highlighted in red, while topic-related entities are highlighted in blue. Given the KG containing all these entities, we intend to generate the description about "*New Jersey Symphony Orchestra*". Firstly, MGCN+SUM is able to cover the main entity and most topic-related entities correctly, while GCN fails to identify the

Gold	The New Jersey Symphony Orchestra is an American symphony orchestra based in the state of New Jersey. The NJSO is the state orchestra of New Jersey, performing concert series in six venues across the state, and is the resident orchestra of the New Jersey Performing Arts Center in Newark, New Jersey.
GCN	The Newark Philharmonic Orchestra is an American orchestra based in Newark, New Jersey, United States.
MGCN +SUM	The New Jersey Symphony Orchestra is an American chamber orchestra based in Newark, New Jersey. The orchestra performs at the Newark Symphony Center at the Newark Symphony Center in Newark, New Jersey.

Table 5: An example of generated sentences.

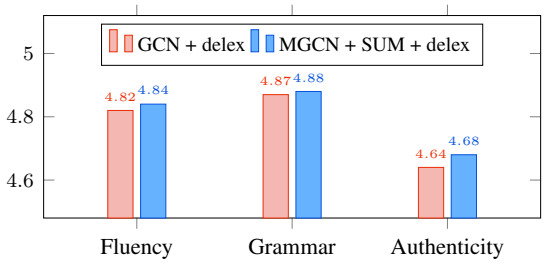


Figure 6: Results for human evaluation.

main entity. This suggests that without multi-graph transformation or effective aggregation methods, it is hard for GCN to extract useful information given a large number of triples in the KG. Length-wise, the output generated by MGCN+SUM is relatively longer than the one generated by GCN, and thus covers more information. We attribute the reason to GCN's deficiency of information loss, as mentioned earlier.

Human Evaluation In order to further assess the quality of the generated sentences, we conduct human evaluation by randomly selecting 100 sentences from outputs generated by GCN+delex and MGCN+SUM+delex. We hire 6 annotators to evaluate the quality based on three evaluation metrics: fluency, grammar and authenticity. In terms of authenticity, annotators rate this metric based on the KG (i.e., Wikidata). More specifically, we give our annotators all main entities' neighbors, 1-hop and 2-hop connections between main entities and topic-related entities as references. A full score will be given if the statements in the generated sentences are consistent with the facts shown in the KG. All three metrics take values from 1 to 5, where 5 states the highest score. The results are shown in Figure 6. Recall that BLEU scores of GCN+delex and MGCN+SUM+delex are 28.4 and 30.0 respectively, we can see from Figure 6 that MGCN+SUM+delex only performs slightly better than GCN+delex on the two language quality metrics, namely, fluency and grammar. For authenticity, the improvement is more significant. Plausibly it is because the 1.6 BLEU improvement results in more impact on the factual correctness.

Models	**BLEU**
TILB-SMT (Gardent et al., 2017)	44.28
MELBOURNE (Gardent et al., 2017)	45.13
MGCN	**45.79**
MGCN + CNN	45.83
MGCN + AVG	**46.55**
MGCN + SUM	45.23

Table 6: Results on WebNLG dataset.

6.4 Additional Experiments

To examine our model's efficacy on a dataset of different characteristics, we conduct an auxiliary experiment on WebNLG (Gardent et al., 2017), which shares the most similarity with ENT-DESC dataset among those benchmark datasets (e.g., E2E, AGENDA, WIKIBIO, etc.). The experiments on WebNLG dataset are under the same settings as the main experiments on our ENT-DESC dataset.

As shown in Table 6, we observe that our proposed models outperform the state-of-the-art model MELBOURNE. However, the performance improvement is less obvious on this dataset, largely due to different characteristics between WebNLG and ENT-DESC. As mentioned in the dataset comparison, the input graphs in WebNLG dataset are much simpler and smaller, where all the information is useful for generation. Our MGCN model would show stronger advantages when applied to a larger and more complicated dataset (e.g. ENT-DESC dataset), where extracting more useful entities and relations from the input graphs and effectively aggregating them together is more essential.

7 Conclusions and Future Work

We present a practical task of generating sentences from relevant entities empowered by KG, and construct a large-scale and challenging dataset ENT-DESC to facilitate the study of this task. Extensive experiments and analysis show the effectiveness of our proposed MGCN model architecture with multiple aggregation methods. In the future, we will explore more informative generation and consider applying MGCN to other NLP tasks for better information extraction and aggregation.

References

Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. 2018. N-gcn: Multi-scale graph convolution for semi-supervised node classification. *Uncertainty in Artificial Intelligence*.

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

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of EMNLP*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of ACL*.

Tianqi Chen, Mu Li, MinLin YutianLi, MinjieWang NaiyanWang, BingXu TianjunXiao, and ZhengZhang ChiyuanZhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint*.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of ACL*.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. 2010. Natural language interfaces to ontologies: combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of international conference on The Semantic Web: research and Applications-Volume Part I*.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the sixth workshop on statistical machine translation*.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of ACL*.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of NIPS*.

Zihao Fu, Lidong Bing, and Wai Lam. 2020a. Open domain event text generation. In *Proceedings of AAAI*.

Zihao Fu, Bei Shi, Wai Lam, Lidong Bing, and Zhiyuan Liu. 2020b. Partially-aligned data-to-text generation with distant supervision. In *Proceedings of EMNLP*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of INLG*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *TACL*.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Technical report.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of ACL*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of ACL*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of EMNLP*.

Juntao Li, Lidong Bing, Lisong Qiu, Dongmin Chen, Dongyan Zhao, and Rui Yan. 2019. Learning to write stories with thematic consistency and wording novelty. In *Proceedings of AAAI*.

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

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of INLG*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of SIGdial Meeting on Discourse and Dialogue*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of EACL*.

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint*.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *Proceedings of ICML*.