Knowledge Graph Empowered Entity Description Generation

Liying Cheng[*1,2], **Yan Zhang**[†2], **Dekun Wu**[2,3], **Zhanming Jie**[‡2], **Lidong Bing**[1], **Wei Lu**[2], **Luo Si**[1]
[1] DAMO Academy, Alibaba Group
[2] Singapore University of Technology and Design
[3] York University
{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

Existing works on KG-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, because the output description may only want to cover the most significant knowledge. In this paper, we introduce a large-scale and challenging dataset to facilitate the study of such practical scenario in KG-to-text. Our dataset involves exploring large knowledge graphs (KG) to retrieve abundant knowledge of various types of main entities, which makes the current graph-to-sequence models severely suffered from the problems of information loss and parameter explosion while generating the description text. 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 ensemble 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 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 (Snow et al., 2008). 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.

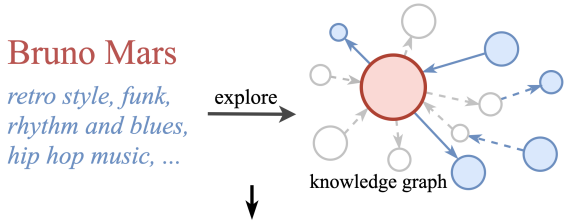


Figure 1: An example showing our proposed task.

To this end, we tackle a knowledge graph empowered entity description generation task 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

[*] Liying Cheng is under the Joint PhD Program between Alibaba and Singapore University of Technology and Design.
[†] Yan Zhang was an intern at Alibaba.
[‡] Zhanming Jie was an intern at Alibaba.
[1] Our code and data are available at https://github.com/LiyingCheng95/EntityDescriptionGeneration.

knowledge from a KG for more natural description. In order to facilitate the study, we introduce a new dataset namely *entity-to-description* (ENT-DESC) 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 how to find the underlying relations between the entities and keywords. In our dataset, we explore such relation revealed in a KG, i.e. the upper right in Figure 1. Therefore, to some extent, our dataset is a generalization of existing KG-to-text datasets in two major aspects. First, 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. Second, our dataset allows the direct generation from the input entities to the output descriptions, and the intermediate knowledge could be explored freely from other sources.

Our proposed dataset is not only more practical but also more challenging due to lack of explicit alignment between input and 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 the problem of low training speed and parameter explosion, as edges are encoded in the form of parameters. Previous work deals with the problem by transforming the original graphs into Levi graphs (Beck et al., 2018). However, Levi graph transformation only represents the relations between original nodes and neighbour edges while relations between original nodes are learned implicitly through 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 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 ensemble the useful information from the KG. Extensive experiments are conducted on both our proposed dataset and benchmark dataset (i.e. WebNLG). Our MGCN outperforms several strong baselines, which demonstrate the effectiveness of our techniques especially when using fewer GCN layers.

The main contributions of this paper can be summarized as follows:

- We tackle a more practical task of knowledge graph empowered entity description generation. We also construct a large-scale dataset ENT-DESC for this specific task. To the best of our knowledge, ENT-DESC is the largest dataset of KG-to-text generation.
- We propose a multi-graph structure transformation that explicitly expresses more comprehensive and more accurate graph information, in order to overcome Levi graph's limitations.
- Thorough experiments and analysis on our new dataset show that our proposed MGCN model incorporated with aggregation methods outperforms strong baselines by effectively capturing and ensembling multi-graph information.

2 Related Work

To better position our work, we first review related tasks and datasets, followed by the works on graph-to-sequence modeling.

Dataset and Task. There are 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 triples (no more than 7) containing equivalent information. Koncel-Kedziorski et al. (2019) introduced AGENDA dataset which aimed to generate paper abstract from a title and a KG. They built the KG by using information extraction system on the abstracts. In our work, we directly explore the Wikidata as our KG without looking at the output. Scale-wise, our dataset consists of 110k instances while AGENDA is 40k. Lebret et al. (2016) introduced WIKIBIO dataset that generated the first sentence of biographical articles with the key-value pairs extracted from the article's infobox. Dušek et al. (2020) 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 from single domain, while ours focuses on multiple domains including humans, events, locations, organizations with more than 100 categories. 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 our proposed KG-to-text task.

Graph-to-sequence Modeling. In recent years, graph convolutional networks (GCN) have been applied to several NLP tasks (e.g. semi-supervised node classification (Kipf and Welling, 2016), 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. 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 Perez-Beltrachini (2018) then used GCN as the encoder in order to capture more distant information in graphs. Since there are usually large amount of labels for edges in KG, such graph-to-sequence models will incur in 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 as mentioned above.

3 Task Description

In this paper, we tackle a practical problem of KG empowered entity description generation. 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 mind, 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, people are aware of describing *Bruno Mars* in the direction of music styles on top of their basic information.

	WebNLG	AGENDA	ENT-DESC
# instances	43K	41K	110K
Input vocab	4.4K	54K	420K
Output vocab	7.8K	78K	248K
# distinct entities	3.1K	297K	691K
# distinct relations	358	7	957
Avg. # triples per inst.	3.0	4.4	27.4
Avg. # words per inst.	23.7	141.3	31.0

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

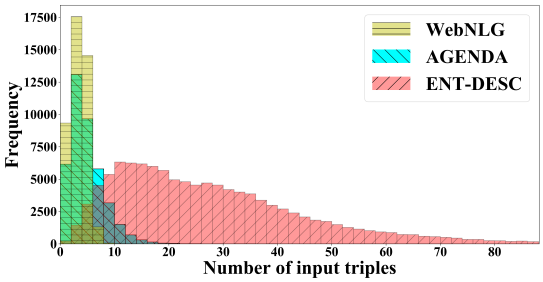
Formally, given a set of entities $\mathbf{e} = \{\mathrm{E}_1, ..., \mathrm{E}_n\}$ and a KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where E_1 is main entity, $\mathrm{E}_2, ..., \mathrm{E}_n$ are topic-related entities, $\mathcal{V}$ is the set of entity nodes and $\mathcal{E}$ is the set of directed relation edges. We intend to generate a natural language text $\mathbf{y} = \{y_1, y_2, \cdots, y_T\}$ which should cover as many entities in $\mathbf{e}$ as possible. Meanwhile, we explore $\mathcal{G}$ for useful information to improve the naturalness of the 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 $\mathrm{V}_{\mathrm{O}_i}$.

4 ENT-DESC Dataset

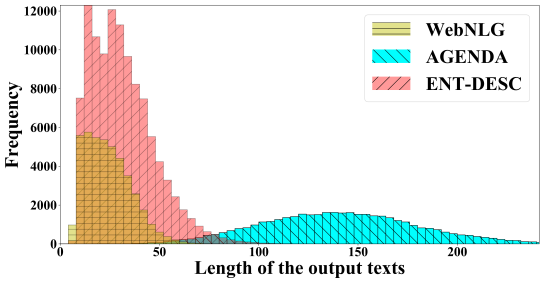
To prepare our dataset, we first use Nayuki's implementation[3] to calculate the PageRank score for more than 9.9 million Wikipages. We then extract the categories from Wikidata for the top 100k highest scored pages and manually select 90 categories out of the top 200 most frequent ones as the seed categories.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

[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 and our ENT-DESC.

hyperlink as topic-related entities. We then search Wikidata to gather neighbours 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. The comparison of our dataset with WebNLG and AGENDA is shown in Table 1 and Figure 2.

In the comparison of these three datasets, there are some obvious differences. First, our dataset is significantly larger than WebNLG and AGENDA (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 sequence. And as observed, quite a portion of text information cannot be directly covered by the input triples. Considering the construction details of both WebNLG and AGENDA, all their input triples provide useful information (i.e. should be used) for generating the output. In contrast, our dataset has 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 better output sequence.

5 Our MGCN Model

In practice, our proposed task can be cast as a problem of generating text from KG. Following most graph-to-sequence generation work, we present an encoder-decoder architecture by introducing multi-graph transformation incorporated with effective aggregation methods, 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 of MGCN, 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 we introduce 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, 2016; 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 neighbours $\mathcal{N}(\cdot)$, 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 directions: (1) v_i to v_j, (2) v_j to v_i, (3) v_i to itself when i equals to 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.

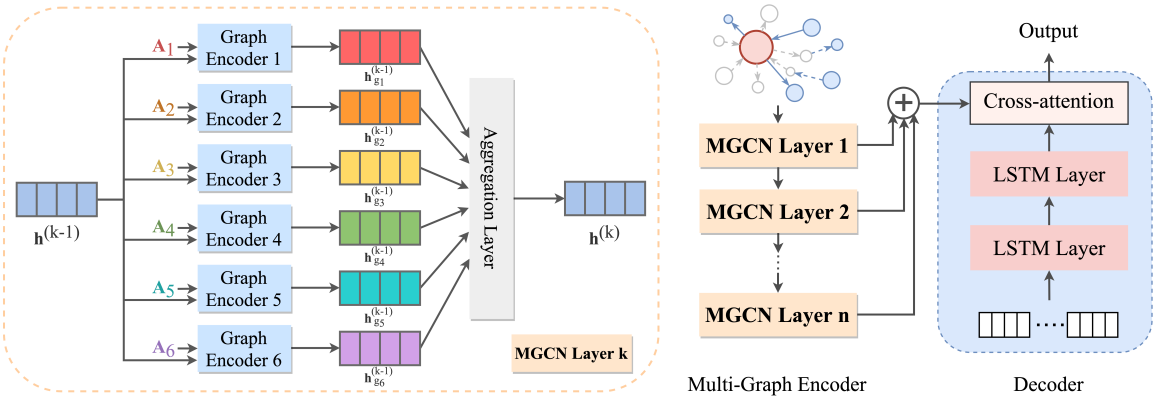


Figure 3: Overview of our model architecture. There are n MGCN layers in the multi-graph encoder, and 2 LSTM layers in the decoder. $\mathbf{h}^{(k-1)}$ is the input graph representation at Layer k, and its 6 copies together with the corresponding adjacent matrices $\mathbf{A}_i$'s of transformed graphs 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.

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 can 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 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 fails to learn different importance of various edge types, and also involves too many features and parameters.

In order to address the challenges mentioned above, we propose three aggregation methods for the multi-graph structure: sum-based, average-

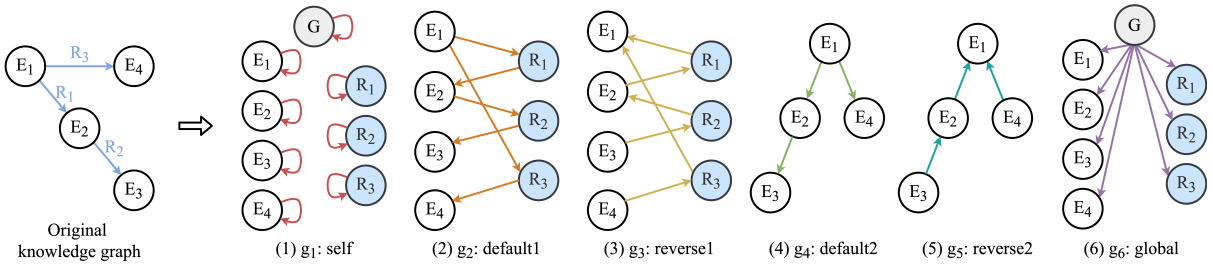


Figure 4: An example of multi-graph transformation.

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 neighbours by stacking multiple MGCN layers. Inspired from 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 follow:

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

Such 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 every 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. Therefore, 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 tookit. 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 batch size of 16. The training phase is stopped when detecting the convergence of perplexity on the validation set. During the decoding phase, we use beam search with beam size of 10.

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), CHRF++ (Beck et al., 2018), METEOR (Denkowski and Lavie, 2011), TER (Snover et al., 2006), ROUGE_1, ROUGE_2, ROUGE_L (Lin, 2004). We adapt Mul-

Models	BLEU	CHRF++	METEOR	TER↓	$ROUGE_1$	$ROUGE_2$	$ROUGE_L$
S2S (Bahdanau et al., 2014)	6.8	24.3	10.8	80.9	38.1	21.5	40.7
GraphTransformer (Koncel-Kedziorski et al., 2019)	19.1	40.0	16.1	94.5	53.7	37.6	54.3
GRN (Beck et al., 2018)	24.4	41.3	18.9	70.8	54.1	38.3	55.5
GCN (Marcheggiani and Perez-Beltrachini, 2018)	24.8	41.7	19.3	70.4	54.9	39.1	56.2
DeepGCN (Guo et al., 2019)	24.9	41.7	19.3	70.2	55.0	39.3	56.2
MGCN	**25.7**	**42.6**	**19.8**	**69.3**	**55.8**	**40.0**	**57.0**
MGCN + CNN	**26.4**	**43.8**	**20.4**	69.4	56.4	40.5	**57.4**
MGCN + AVG	26.1	43.4	20.2	**69.2**	56.4	40.3	57.3
MGCN + SUM	**26.4**	43.6	20.3	69.8	56.4	**40.6**	**57.4**
GCN + delex	28.4	48.5	22.9	65.9	61.8	45.5	62.1
MGCN + CNN + delex	29.6	50.0	**23.7**	**63.2**	**63.0**	**46.7**	**63.2**
MGCN + SUM + delex	**30.0**	**50.1**	**23.7**	67.4	62.6	46.3	62.7

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

tEval (Clark et al., 2011) and Py-rouge for resampling and significance testing.

6.2 Main Experimental Results

Here we present our main experiments on our prepared ENT-DESC dataset. We compare our proposed MGCN models with various aggregation methods against several strong GCN baselines including sequence-to-sequence (S2S) (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), and report the results on the test set. We re-arrange the order of input triples following the occurrence of entities in output for S2S model. We re-implemented GRN, GCN and DeepGCN using MXNET. Furthermore, we apply a delexicalization technique on our dataset. Unlike previous delexicalization work by applying name entity anonymization (Konstas et al., 2017), we delexicalize the main entity and topic-related entities by replacing these entities with a token indicating the entity type and index.

Main results on our ENT-DESC dataset are shown in Table 2. Here, number of layers in all baseline models and our MGCN models are set to be 6 for fair comparison. 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 demonstrates the effectiveness of multi-graph transformation, which is able to capture more comprehensive information compared to Levi graph transformation, especially entity-to-entity information in the original graph. We then apply multiple methods of aggregation on top of multi-graph structure. MGCN+CNN and MGCN+SUM report the highest BLEU score 26.4, followed by MGCN+AVG. By applying our delexicalization technique, the results are further boosted by 3.2 to 3.6 BLEU score for both baseline and our proposed models.

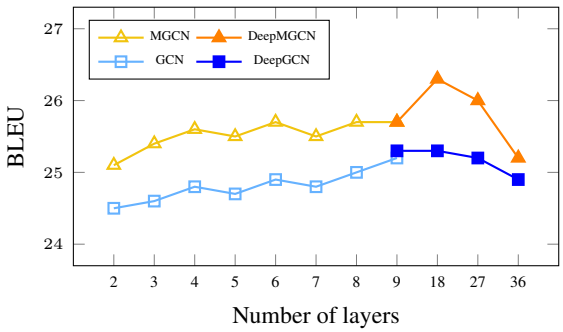


Figure 5: Effect of MGCN/deepMGCN on different number of layers.

6.3 Analysis and Discussion

Effect of different numbers of MGCN layers. In order to examine the robustness of our MGCN models, we conduct contrast 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 model on 2 to 9 layers and DeepGCN on 9, 18, 27 and 36 layers. Both models perform better initially as more GCN/MGCN layers are being stacked, and start to drop afterwards. DeepMGCN, achieving 26.3 BLEU score at 18 MGCN layers, is 1.0 BLEU higher than deepGCN model. As shown in the line chart (Figure 5), MGCN achieves a decent increase of 0.3 to 1.0 from 2 to 36 layers. This again shows robust improvements by explicitly representing the all types of information in the graph than learning the information implicitly. Another observation is that the BLUE of MGCN model at 3 layers (25.4) is already higher than the best performance of GCN/deepGCN model regardless of number of layers.

# 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 MGCN+SUM on 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.

Effect of MGCN+SUM on various numbers of input triples. In order to have a deeper understanding on 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 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. With small amount of triples, the models have insufficient information to generate accurate descriptions while large amount of triples make it challenging for the models to select meaningful information. Another observation is that the improvement of BLEU (Δ) by our model is higher with smaller number of triples.

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, 2016). We notice that the result drops after removing any transformed graph from the multi graph. Particularly, we observe that the importance of $\{default2, reverse2\}$ and $\{default1, reverse1\}$ are equivalent, as the BLEU 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 reverse direction for more effective information extraction in directed graphs as there are relatively larger gaps in BLEU drop after removing g_3 (*reverse1*) and g_1 (*reverse2*).

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.

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 model fails to identify the main entity. This suggests that without multi-graph transformation or effective aggregation methods, it is hard for GCN to extract useful information given large number of triples in the KG. Lengthwise, 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.

7 Conclusions and Future Work

We presented a practical task of generating sentences from relevant entities empowered by KG. We further constructed a large-scale and challenging dataset ENT-DESC to facilitate the study of this task. To overcome the limitations of previous graph-to-sequence models on large KGs, we proposed a multi-graph convolutional networks (MGCN) incorporated with multiple aggregation methods to capture the rich structured information in the KG. Thorough experiments and analysis show the effectiveness of our model architecture. In the future, we will consider incorporating pretrain knowledge for more informative generation, and also explore applying MGCN on 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.

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

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.

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 the 7th 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*, pages 85–91. Association for Computational Linguistics.

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*.

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 *Advances in neural information processing systems*, pages 2224–2232.

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*, 7:297–312.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8).

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

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks.

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*.

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, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. ArXiv:1706.09254.

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. *EACL*, page 157.

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*, volume 200.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*.

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

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*, pages 5449–5458.