Research Proposal
High-quality Graph-to-Text generation

Yinhong Liu

***Abstract*— Natural Language Generation has undergone significant advancement in the recent past. In Data-to-Text generation, significant progress has been made in coherency and grammatical fluency of the generated texts thanks to prevalent neural encoder-decoder structure. However lack of inter-sentence structure of the automatically generated multi-sentence text has been a long-existing problems. In this proposal, I address the problem of generating coherent, logical and faithful multi-sentence texts from structured input data, in particular, a Knowledge Graph(KG). Based on latest Graph-to-Text generation model, the GraphWriter [1], and others, I propose two potential directions to improve it: Integrating with node embedding and modeling content plan as a separated stage.**

I. Introduction

A. *Motivation to This Work*

The progress of Artificial Intelligence can be abstractly considered in two main stages, perception and cognition. Perception is the ability to observe patterns, be aware of representations of things and learn their regularities, while cognitive intelligence usually refers to the ability to handle logical reasoning, solving abstract problems and comprehending complex ideas. In recent years, although there are substantial progress in many machine learning research areas, especially in image processing and computer vision, these advances mainly fall into the artificial perception area. However, most Natural Language Processing tasks are intrinsically trying to model cognition as they usually involve understanding and comprehension. To understand succinct natural utterances, large amount of latent information is required by the machine. For example, to generate sentence like 'Where can I find a convenient store? I'm thirsty,' the machine needs to first understand the background information that 1) Thirst implies the need for drinks, and 2) a convenient store has drinks.

Knowledge Graph (KG) is a kind of database that contains abundant detail information and graph is an ubiquitous data structure in computing. KG is usually considered to be more flexible and efficient than other types of database, like tables and trees. Therefore, there are many Data-to-Text research trying to generate text using graphical input or triples, such as [1] and [2].

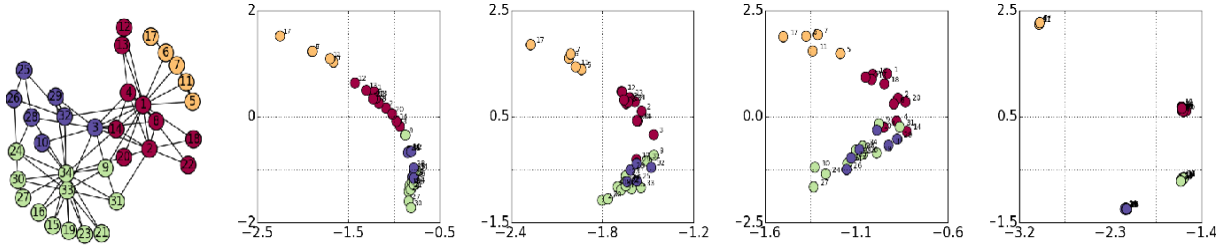
An ideal Graph-to-Text generation model could have substantial influence in both academic and industrial aspects. Academically it provides a fundamental way for machine to communicate with human more naturally, which means it could have further influence in areas like human-machine interaction. In industry, there are massive existing sources of Knowledge Graphs across many domains, like social networks by social media companies, road system networks by governments and ground-truth/common-sense database by Wikipedia. An ideal Graph-to-Text model can leverage those data to automatically generate pragmatic text that can be used in extensive aspects of a society, ranging from scientific report reasoning to interactive-conversation generation.

However, similar to other Data-to-Text generation models, even the latest Graph-to-Text model still suffers from problems such as lack of logic, selection of content and faithfulness, when generating multi-sentence text (which will be explained in the following section I-B). As the output text is directly used for reading and human reader usually has high expectation on the text quality, lots of attention is attracted to confront the aforementioned difficulties. This proposal will target on these problems and discuss potential methods to improve them based on latest research achievement.

B. *Problem Setting Breakdown*

Traditional approach [3] [4] to Data-to-Text generation task usually followed a pipeline of procedures which include content planning (selecting specific content from the input and planning the structure of the output text), micro planning (determining the concrete structure of each sentence) and sentence realization (generating natural sentence from the sentence plan). These modules can produce text that is well-structured and faithful to the input data, as they are usually implemented by template-based methods, which also means they are intrinsically inflexible and cannot leverage large-scale data.

Recent popular neural network approaches tend to not explicitly model any of these stages. Instead, they tend to use the encoder-decoder structure with an end-to-end training fashion. The results are successful: many models have shown extraordinary improvements in generating grammatically fluent text. However, compared to the traditional template-based methods, neural network methods are not as good at capturing long-distance logic or dependencies, especially for input data represented in graph as the non-hierarchical and unordered nature makes long distance information even more intractable. As as result, the generated text usually suffers problems of lack of inter-sentence coherence, generating redundant or



(a) Karate club graph (b) DeepWalk (c) LINE (d) node2vec (e) ComEmbed (ours)

Fig. 1: Node embedding by different methods. (This figure is credited to from paper [5].)

repeated information and may even be unfaithful to the input.

In this proposal, the problem of lack of structure in multi-sentence Graph-to-Text generation is addressed. I will discuss the potential of integrating node embedding learning with the graph transformer in GraphWriter[1], as by this way text attributes along with structural information can be successfully encoded in to node embedding. In addition, I will also investigate potential methods to bring content plan back, as a separated stage, to the current popular neural generation model. Specifically, explicit content plan modeling and how to generate it from graph input will be discussed.

II. Related work

The Graph-to-Text generation is a part of larger concept Data-to-Text generation. In the early stage, research [6] has been studied in Natural Language Generation with pipeline, which includes content planning, micro planning and realization. Before neural approaches became prevalent, there is a period of various statistical models, including generative modeling [7], end-to-end probabilistic approach [8] and probabilistic context-free grammar [9], [10]. Although these methods have hierarchical structure during generation, they are usually partially template-based, which means rules need to be specified at every decision level. In the recent years, neural models that use successful attention-based encoder-decoder structure have achieved a new level of Data-to-Text generation. Most of them choose to abandon the content planning stage. As a representative, the attention-based sequence-to-sequence model [11] takes tabular text input and directly outputs word sequences.

It was not until recently that some researcher starts to explicitly model the content plan as a part of the end-to-rated stage. In research [12], it models the content selection and planning as weightings and orderings of the input tabular record. In paper [13], the content planning is split from the generation model and modelled as a sequence of trees, using template-based method. The GraphWriter model [1], on which this proposal is heavily based, proposed a graph transformer structure that implicitly model the planning by considering attentions from surrounding nodes. This proposal will focus on the potential improvement of this model and further.

In addition, this proposal will also discuss the way graph representation learning can help to improve the current Graph-to-Text generation. There are many successful node/graph representation learning models in its long research history. Early matrix-factorization based approaches [14], [15] are directly derived from concept of dimensionality reduction. Random-walk based methods [16], [17] can give the node embedding in a way that if nodes tend to co-occur on short walking distance in the graph, they usually have similar embeddings. In the recent years, neural-based models have become more popular and successful due to their ability to leverage large-scale data. Research like Deep Neural Graph Representation (DNGR) [18] and Graph Convolutional network (GCN) [19] have adopted the encoder-decoder structure in attempt to take more textual and structural information from neighbour nodes.

III. Methodology

As mentioned, I will discuss the potential improvements of Graph-to-Text generation in two ways based on the current GraphWriter [1] model. The first attempt is to integrate node representation learning to the graph transformer structure, so that during training, each node can learn attentions of nodes from further distance. In another word, more structural information can be encoded into the node embedding. The second idea is to model the content planning as a separated stage, so that in the second stage, the model can focus more on grammatically coherence and therefore give better realization.

A. *GraphWriter with Node Embedding*

The GraphWriter [1] model designs a novel self-attention learning structure which adopts the latest and successful transformer structure. However, in my opinion, the way the graph transformer is used has a few limitations:

- Entities of nodes are usually made up of multi-word phrases. The method used to encode the entity is first generating word embedding with some pre-trained embedding matrix. Then the sequence of embedded word vectors is passed to a bi-directional RNN, the output of which is then used as the representation of the node entity. By this way, the embedding only consider the text attribute of a node, but not any structural

information, which could be a waste of the informative graph structure.

- For each self-attention layer in the graph transformer, as every node only looks at attentions from its neighbour nodes and one global node. This global node design is similar to the global node in node embedding learning section of the famous Graph2Seq [20] model, which is used to provide the node its positional information in the graph. However, if we do not consider the effect of global node (as only limited information can be conveyed by a single node), in each self-attention layer, information can only be passed by one hop, which is rather inefficient when the graph is large.

To cope with the above drawbacks, we can adopt the idea of learning the node representation. The concept of node embedding is extended from and thus has similar meaning with the successful word embedding pretraining, like Word2Vec [21], GloVe [22] (Although they suffer problem of word ambiguity, node embedding tends to be immune to it, as meanings of node entities are usually unambiguous). Prior contextual and positional knowledge of the nodes can be encoded into the embedding. As shown in Fig.1, there are extensive methods of learning node representations, including matrix-factorization based methods, random-walk based methods and attention-based mechanisms. They share a basic concept that is to map the node from the graphical space to low dimensional space in a way that similar nodes should locate close to each other. Different methods are different in the way they map the node into embedding space and the way they define their similarity functions.

- There are various similarity functions: adjacency-based similarity values the closer neighbours more. Multi-hop similarity gives similar weights to the nodes that are equal-hop away from target node. Random-walk-based methods then believe nodes that are likely co-occurring on a random walk over the network should be closer in embedding space. Different similarity functions might be suitable for different datasets, therefore it is worth examining which one can play a better role in the graph transformer given data.
- We can take one step further and investigate the possibility to adopt a novel embedding training objective: rather than choosing an existing similar function, we deliberately mask some nodes on the graph and use knowledge graph completion task to try to predict the masked nodes, just like the training objective of the language model in BERT [23].

However, no matter which node embedding method we choose, there are a few potential problems:

- For the case of the nodes, those of KG are made up of multi-word phrases, which tends to be sparse in nature. However, it is hard to learn meaningful representations for those nodes, as they are usually connected to very few other nodes in their KG. In this case, the text attribute will play dominant role in the node embedding.

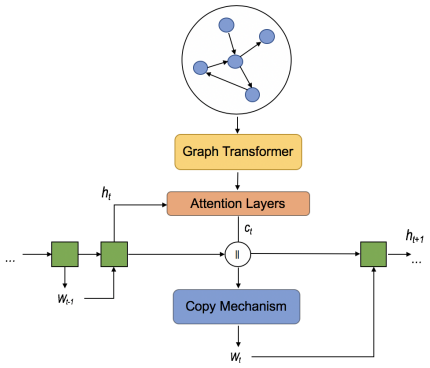


Fig. 2: Graph-to-Sequence content plan generation. (graph is modified based on the GraphWriter [1])

- Spellings of words can have variants, same as the node entities. However, the variants of word usually contain latent information like plurality and tense, but for entity phrases, the variants are usually just spelling issues, which makes them even sparser in their KG.
- The knowledge domain of the input KG can also cause sparsity problem. KG naturally contains large amount of trivial details and relations, and therefore if the input KGs are not restricted within a tight domain, many nodes will only occur very few times in the dataset.

The way to counter the above issues is simple but expensive: we need the dataset to be large in scale and consistent in node entity expressions.

B. Generation with Separated Content Plan Modeling

In the recent years, Data-to-Text generation methods generally adopt prevalent neural model with an end-to-end training fashion. They abandon the content planning stage and let neural network learn everything in one-shot. However, as aforementioned, the text generated in this way suffers problems of lack of inter-sentence structure, content selection and even faithfulness. In latest research, there are some attempts on integrating content planning as a separated stage with the neural model. However, the existing models either relaying on template-based methods to generate the content plan, like [**?**], or requiring input to be in tabular format [12]. So far, research in integrating content plan modeling for Graph-to-Text generation remains blank.

We first need to model the content plan with a data structure. I proposed two candidate structure, sequence of nodes or sequence of sub-graph. In the following section, I will explain how to generate content plan in both structures from graphic input and potential problems they may have.

*1) **Sequence of nodes**:* By modeling the content plan as a sequence of nodes (node here is expressed as node vector), we can effectively split the original generation task into two parts. Graph-to-Sequence for plan generation and then Sequence-to-Sequence for sentence realization. There are extensive successful models in Sequence-to-Sequence

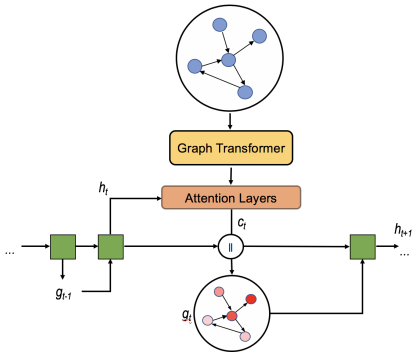


Fig. 3: Graph-to-Graph content plan generation. (graph is modified based on the GraphWriter [1])

generation, so in this proposal I will mainly focus on the Graph-to-Sequence stage.

The GraphWriter [1] has provided a novel and potential graph transformer structure for graph understanding. Therefore, we can use it in a way that in each decoder step, the output can only be the nodes copied from the knowledge graph, as shown in Fig.2. There are many copy mechanisms, apart from the copy mechanism used in GraphWriter. We can try out other mechanisms [24] [25], to see which one is more suitable for our Graph-to-Sequence model. By this manner, we can output a specific node of the input KG in each step, which is also one step of our content plan. Then we can map each node to what we want to pass to the realization stage, which could be, as an example, an aggregation of all attentions from neighbour nodes or others. The details of the mapping process remain open to investigation.

*2) **Sequence of sub-graph**:* We can also model the content plan as a sequence of sub-graphs of the KG. A sub-graph could be expressed as a masked node matrix of the input KG. By this way, the generation task becomes a Graph-to-Graph + Graph-to-Sequence problem. As again, we can apply the GraphWriter model in the Graph-to-Sequence realization stage, we will only focus on the first plan generation stage. As shown in Fig.3, we can modify the decoder of the GraphWriter so that in each step of the output it will output weightings for all nodes in the graph. Then we can apply a selection mechanism, e.g., applying a threshold to the weightings, to determine which nodes should be in the output sub-graph. The representations and weightings of the nodes in the selected sub-graph will be input to each step of encoder in the second stage.

*3) **Problems**:* The above proposed models are rather optimistic and in reality, there will be various problems implementing them. In this section, I will discuss about the potential problems and their solutions.

- Splitting the content plan as a separated stage and explicitly modeling it will cause a fundamental change from the original model: The end-to-end training manner can no longer be maintained and therefore, for current Graph-to-Text datasets, we need to generate content plans(in the format of either of the proposed data structures) as an intermediate reference. There is no existing method to do this and manually generation could be very expensive.
- In the decoder of the first proposed method III-B.1, although we model the plan as a sequence of nodes, we cannot only input a single node vector to the second realization stage. Instead, we need some mechanism to consider some information from neighbouring nodes, for example, an aggregation of attentions from surrounding nodes. Adding to this, we also need to determine which neighbouring nodes should be taken into account. Therefore, more detail investigation is required in this open question.
- In the second proposed method III-B.2, given the weightings of every node, the selection mechanism could be complicated and potentially affect the model performance in a great extent. Different selecting rules might cause different problems, for example, the aforementioned simple threshold method will potentially select nodes that are distant in the KG, which will give a hard time for the realization model to understand it. Therefore, similarly, this open area is also worthy to study.
- Last but not least, as the content plan is explicitly generated, we need a way to evaluate it. However, rating different content plans is a fairly subjective decision. Even given same information, people could make their expression in completely different ways. There is hardly a legitimate way to score them and for most of the cases. Adding to this, if we specify a rating rule, we will restrict all other possible varieties, which is not preferable. Therefore, further research is needed to solve this fundamental problem of content plan generation. One potential solution is to have more than one intermediate reference for each graph-text data pair, which will make datasets even more expensive.

IV. Conclusion

In conclusion, my interest falls into Data-to-Text, especially Graph-to-Text, generation and I believe progress in this area has both academic and pragmatic values. The frontier of this research, in my opinion, is the novel graph transformer design in the GraphWriter model [1]. In this proposal, I suggested two potential directions to improve this model. The first one is to utilize current node representation learning methods, so that more structural information can be learned in the graph transformer. The second suggestion is trying to bring back the traditional content planning stage to the currently popular neural encoder-decoder models by explicitly generating the plan in separated encoder-decoder structure. The drawbacks of the proposed methods can be concluded into two aspects: Requiring datasets to have content plans as intermediate references and requiring large scale of in-domain data.

REFERENCES

[1] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text Generation from Knowledge Graphs with Graph Transformers," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 2284–2293, Association for Computational Linguistics, June 2019.

[2] B. D. Trisedya, J. Qi, R. Zhang, and W. Wang, "GTR-LSTM: A triple encoder for sentence generation from RDF data," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 1627–1637, Association for Computational Linguistics, July 2018.

[3] E. Reiter and R. Dale, *Building Natural Language Generation Systems*. New York, NY, USA: Cambridge University Press, 2000.

[4] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *CoRR*, vol. abs/1703.09902, 2017.

[5] V. W. Zheng, S. Cavallari, H. Cai, K. C. Chang, and E. Cambria, "From node embedding to community embedding," *CoRR*, vol. abs/1610.09950, 2016.

[6] E. Reiter, *Building Natural Language Generation Systems*. Cambridge Unicersity Press, 2000.

[7] P. Liang, M. I. Jordan, and D. Klein, "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: Volume 1 - Volume 1*, ACL '09, (Stroudsburg, PA, USA), pp. 91–99, Association for Computational Linguistics, 2009.

[8] G. Angeli, P. Liang, and D. Klein, "A simple domain-independent probabilistic approach to generation," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, (Stroudsburg, PA, USA), pp. 502–512, Association for Computational Linguistics, 2010.

[9] I. Konstas and M. Lapata, "Unsupervised concept-to-text generation with hypergraphs," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, (Stroudsburg, PA, USA), pp. 752–761, Association for Computational Linguistics, 2012.

[10] I. Konstas and M. Lapata, "Inducing document plans for concept-to-text generation," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, (Seattle, Washington, USA), pp. 1503–1514, Association for Computational Linguistics, Oct. 2013.

[11] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *CoRR*, vol. abs/1509.00685, 2015.

[12] R. Puduppully, L. Dong, and M. Lapata, "Data-to-text generation with content selection and planning," *CoRR*, vol. abs/1809.00582, 2018.

[13] A. Moryossef, Y. Goldberg, and I. Dagan, "Step-by-step: Separating planning from realization in neural data-to-text generation," *CoRR*, vol. abs/1904.03396, 2019.

[14] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, (Cambridge, MA, USA), pp. 585–591, MIT Press, 2001.

[15] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, (New York, NY, USA), pp. 37–48, ACM, 2013.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *CoRR*, vol. abs/1607.00653, 2016.

[17] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *CoRR*, vol. abs/1403.6652, 2014.

[18] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 1145–1152, AAAI Press, 2016.

[19] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pp. 593–607, 2018.

[20] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin, "Graph2seq: Graph to sequence learning with attention-based neural networks," *CoRR*, vol. abs/1804.00823, 2018.

[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (USA), pp. 3111–3119, Curran Associates Inc., 2013.

[22] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[23] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[24] J. Gu, Z. Lu, H. Li, and V. O. K. Li, "Incorporating copying mechanism in sequence-to-sequence learning," *CoRR*, vol. abs/1603.06393, 2016.

[25] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Berlin, Germany), pp. 140–149, Association for Computational Linguistics, Aug. 2016.