How to Train Your Agent to Read and Write

Li Liu,[1,2*] Mengge He,[1*] Guanghui Xu,[1] Mingkui Tan,[1,4†] Qi Wu[3]
[1] School of Software Engineering, South China University of Technology,
[2] Pazhou Laboratory, [3] University of Adelaide,
[4] Key Laboratory of Big Data and Intelligent Robot, Ministry of Education
{seliushiya, semenggehe, sexuguanghui}@mail.scut.edu.cn, mingkuitan@scut.edu.cn, qi.wu01@adelaide.edu.au

Abstract

Reading and writing research papers is one of the most privileged abilities that a qualified researcher should master. However, it is difficult for new researchers (*e.g.,* students) to fully grasp this ability. It would be fascinating if we could train an intelligent agent to help people read and summarize papers, and perhaps even discover and exploit the potential knowledge clues to write novel papers. Although there have been existing works focusing on summarizing (*i.e.*, reading) the knowledge in a given text or generating (*i.e.*, writing) a text based on the given knowledge, the ability of simultaneously reading and writing is still under development. Typically, this requires an agent to fully understand the knowledge from the given text materials and generate correct and fluent novel paragraphs, which is very challenging in practice. In this paper, we propose a Deep ReAder-Writer (DRAW) network, which consists of a *Reader* that can extract knowledge graphs (KGs) from input paragraphs and discover potential knowledge, a graph-to-text *Writer* that generates a novel paragraph, and a *Reviewer* that reviews the generated paragraph from three different aspects. Extensive experiments show that our DRAW network outperforms considered baselines and several state-of-the-art methods on AGENDA and M-AGENDA datasets. Our code and supplementary are released at https://github.com/menggehe/DRAW.

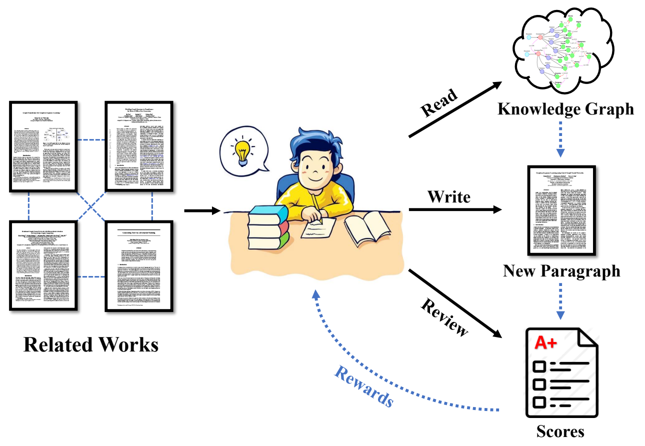

Figure 1: An intuitive understanding of our DRAW network. First, the DRAW network reads multiple related works and discovers potential knowledge among them. And then, it writes a new paragraph based on knowledge graph. Last, it reviews the output and uses feedback rewards to improve the quality of writing.

1 Introduction

Currently, hundreds of papers are published online every day even on small topics. However, a study (Wang et al. 2019) shows that US scientists can only read 264 papers per year on average. Thus, researchers are exhausted by following the sharply increased numbers of papers, much less to understanding the research and coming up with new ideas to write novel papers (Gopen and Ja 1990; Buenz 2019). In practice, writing novel papers requires not only the abilities of reading and reasoning but also the ability of creative thinking, which is nontrivial for most fresh researchers (Xiao et al. 2020). It would be fantastic if an agent could help people, especially new researchers, to read and write. However, building such an agent encounters several challenges.

First, to understand multiple related works, the agent needs to capture complex logic in the related works, which is nontrivial. Several knowledge extraction methods (Min et al. 2006; Gerber and Chai 2010; Yoshikawa et al. 2010) achieve it by identifying entities in the texts, extracting the relationships between these entities, and representing them as a knowledge graph (KG). However, they have trouble in discovering potential connections among these entities, which hampers a comprehensive understanding of related works.

Second, after generating a KG, the agent is then required to decode a fluent novel paragraph from the KG. In practice, however, how to evaluate the quality of the generated texts accurately is still an open problem. Existing methods (Koncel-Kedziorski et al. 2019; Wang et al. 2019) adopt the teacher-forcing scheme that aims to match the tokens in the generated texts to the tokens in the target texts. However, these methods

*Authors contributed equally.
†Corresponding author.

only focus on token-level matching while ignoring sentence-level and graph-level evaluation of the generated texts.

In this paper, we propose a method named Deep ReAder-Writer (DRAW). Our DRAW network is able to read multiple texts, discover potential knowledge, and then write a novel paragraph. From Figure 1, the DRAW network consists of three modules: *i.e., Reader*, *Writer* and *Reviewer*. Specifically, the *Reader* first extracts KGs from the research texts and discovers potential knowledge to enrich the KGs. The *Reader* considers the multi-hop neighborhood to predict new links among conceptual nodes. Then, the *Writer* writes a novel paragraph to describe the main idea of the enriched KGs using a graph attention network, which aggregates the global and local graph information. Inspired by the review process of research papers, we further propose a *Reviewer* module to evaluate the quality of the generated paragraphs and return rewards as feedback signals to refine the *Writer*. To be specific, given a generated paragraph, the *Reviewer* will output three feedback signals, including (1) a quality reward, which reflects the metric scores of the generated paragraph; (2) an adversarial reward, which denotes the probability of the generated paragraph passing the Turing test; and (3) an alignment reward, which represents the matching score between the generated paragraphs and the enriched KGs. In this way, the *Writer* is able to write better paragraphs that clearly represent the key idea of the enriched KGs.

In summary, our main contributions are threefold:

- We propose a Deep ReAder-Writer (DRAW) network that reads multiple research texts and then discovers potential knowledge to write a novel paragraph covering the key idea of the source inputs.
- We propose a feedback mechanism to review whether the generated paragraph is consistent with the enriched KG, and whether the generated paragraph is human written, thereby greatly improving the quality of paragraph generation.
- Extensive experiments show that our *Writer-Reviewer* leads to significant improvements in the KGs-to-text generation task and outperforms the state-of-the-art methods.

2 Related Work

Automatic writing. PaperRobot (Wang et al. 2019) performs as an automatic research assistant to incrementally write to chemical-related research datasets. It enriches KGs by predicting links of input papers' KGs. According to a given title, it then selects several entities that are related to the title in enriched KGs to generate texts. However, PaperRobot neglects to consider the multi-hop neighborhood to predict links, which is very important for capturing potential relationships. In addition, the generated texts do not closely align with the KGs. To address this, we use a graph attention network to consider the multi-hop neighborhood, capturing the complex and hidden information that is inherently implicit in the neighborhood. Moreover, we design a *Reviewer* to measure the quality of the generated text from different dimensions to effectively align with the KGs. In particular, our DRAW network is different from the multi-document summary (Ling and Hui 2013), which compresses the lengthy document content into several relatively short paragraphs. We not only extract important knowledge but also discover potential knowledge from multiple paragraphs by predicting links and writing a novel paragraph.

Link prediction. Some translation-based approaches (Bordes et al. 2013; Zhen et al. 2014; Lin et al. 2015) are widely used in link prediction but result in poor representation ability. Recently, CNN based models (Dettmers et al. 2018; Nguyen et al. 2018) have been proposed for relation prediction. These methods only focus on the entity and its neighborhood while not considering the relationships among these nodes. Other methods (Kipf and Welling 2017; Schlichtkrull et al. 2018) take the relationships among the entities and their 1-hop neighbors into consideration. However, they still omit the information from multi-hop neighborhood. Instead, we propose a *Reader* module to capture semantic information of the multi-hop neighborhood in the KG.

Graph-to-Text task. Graph-to-Text is an active research area. Some works generate texts based on structured knowledge (Trisedya et al. 2018; Xu et al. 2018a; Feng et al. 2018), while several neural graph-to-text models use different encoders based on GNN (Ribeiro, Gardent, and Gurevych 2019; Zhijiang et al. 2019; Huang et al. 2020) and Transformer (Vaswani et al. 2017) architectures to learn graph representations. Koncel-Kedziorski et al. proposes a novel graph transformer encoder, which leverages the topological structure of KGs to generate texts. However, it ignores the global graph information, which is important for text generation. To solve this, Ribeiro et al. introduce a novel architecture that aggregates both global and local graph information to generate texts. However, such an encoder-decoder framework presents some problems such as word repetition and lack of diversity. To solve these issues, we propose a *Reviewer* module to review the generated paragraphs and refine the quality of paragraphs using feedback rewards. Our *Reviewer* consists of three modules to review and evaluate whether the generated paragraphs are real and to align with the given KGs, in order to improve the text generation ability.

3 Proposed Method

In this paper, we focus on generating novel paragraphs via reading multiple AI-related paragraphs. To this end, we propose a Deep ReAder-Writer (DRAW) network that consists of three modules, namely *Reader*, *Writer*, and *Reviewer* as shown in Figure 2. To understand and sort out the textual logic of given paragraphs, the *Reader* first 'reads' and extracts knowledge graphs (KGs) from them. And then, considering the multi-hop neighborhood, the *Reader* predicts new links between conceptual nodes, namely potential knowledge, to enrich the KGs. The *Writer* adopts a graph encoder to encode the rich semantic information in KGs, and delivers it to a text encoder to generate a novel paragraph. Inspired by the adversarial learning (Cao et al. 2019; Wang et al. 2018; Cao et al. 2020; Chen et al. 2020), we also devise a *Reviewer* to evaluate the quality of the generated paragraph, which serves as a feedback signal to refine the *Writer*. We relate the details of these modules in the following sections.

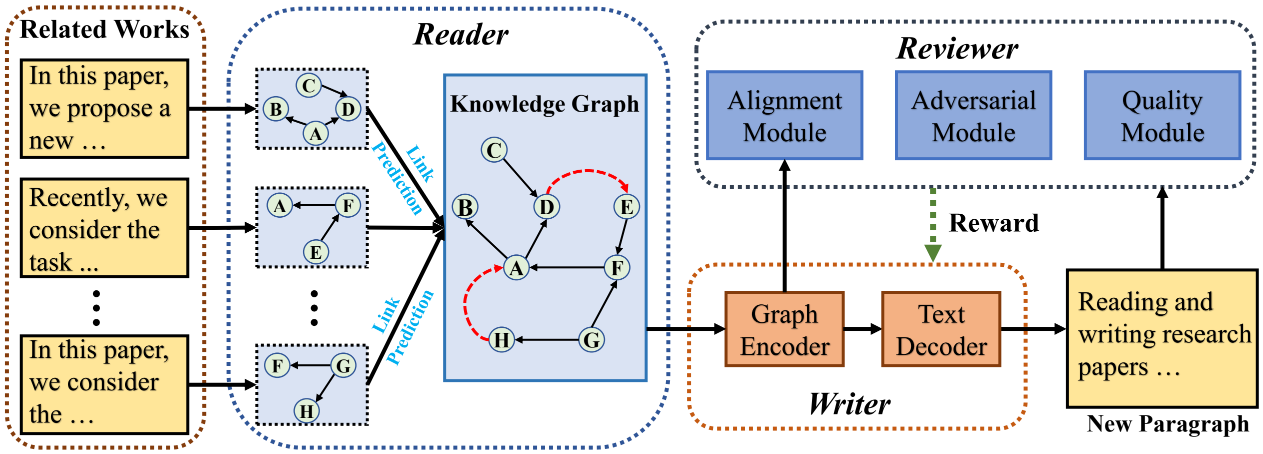

Figure 2: An overview of our Deep ReAder-Writer (DRAW) Network. The DRAW network consists of three modules, namely *Reader*, *Writer* and *Reviewer*. Given multiple related works, the *Reader* first extracts knowledge to construct initial knowledge graphs (KGs) and performs link prediction to enrich KGs. Based on the enriched KGs, the *Writer* captures global and local topology information using a graph encoder and generates a novel paragraph with a text decoder. In particular, the *Reviewer* employs three feedback modules to measure the quality of the generated paragraph.

3.1 Reader: Text-to-Graph

To extract the textual logic from the given related paragraphs, we use the standard SciIE (Luan et al. 2018), a science domain information extraction system to construct knowledge graphs Specifically, the output of the SciIE system is a list of triplets, where each triplet consists of two entities and the corresponding relation. The knowledge graph denoted as $\mathcal{G}_I = \{\mathcal{V}, \mathcal{R}\}$, where $\mathcal{V} = \{\bar{\mathbf{v}}_i\}_{i=1}^N$ is the node set, $\mathcal{R} = \{\bar{\mathbf{r}}_{ij}\}_{i,j=1}^N$ is the edge set and N represents the number of nodes. $\mathcal{V}$ and $\mathcal{R}$ represent the extracted entities and the relations, respectively. However, the initial knowledge graph $\mathcal{G}_I$ does not exploit potential knowledge. To address this, we perform a link prediction to predict new links between entities based on the initial KGs.

Link prediction. Given KGs $\mathcal{G}_I$, we obtain the entity embedding $\bar{\mathbf{v}}_i \in \mathbb{R}^d$ and relation embedding $\bar{\mathbf{r}}_{ij} \in \mathbb{R}^d$ with two separated embedding layers, where d is the feature dimension. Formally, given entity embedding $\bar{\mathbf{v}}_i$, $\bar{\mathbf{v}}_j$ and relation embedding $\bar{\mathbf{r}}_{ij}$ between them, the triplet is represented by $(\bar{\mathbf{v}}_i, \bar{\mathbf{r}}_{ij}, \bar{\mathbf{v}}_j)$. To aggregate more information, we introduce auxiliary edges between one entity and its n-hop neighborhood. For the entity and its n-hop neighborhood, we sum the embeddings of all the relations in the path between them as the auxiliary relation embedding. We apply a linear transformation to update the entity representation $\tilde{\mathbf{v}}_i \in \mathbb{R}^d$ by:

$$\tilde{\mathbf{v}}_i = \mathbf{W}_1[\bar{\mathbf{v}}_i,\ \bar{\mathbf{r}}_{ij},\ \bar{\mathbf{v}}_j], \tag{1}$$

where $\mathbf{W}_1$ is a trainable parameter and $[\cdot, \cdot]$ denotes the concatenation operation. A particular entity $\mathbf{v}_i$ may be involved in multiple triplets and its neighborhood can be denoted as $\{\tilde{\mathbf{v}}_i^k\}$, where $\tilde{\mathbf{v}}_i^k$ denotes the k-th neighborhood of the i-th entity. To learn the importance of each triplet for the entity, we apply a self-attention to calculate attention weights as follows:

$$\widehat{a}_k = \frac{\exp\left(\tilde{\mathbf{v}}_i^k\right)}{\sum_k \exp\left(\tilde{\mathbf{v}}_i^k\right)}, \tag{2}$$

With the help of the attention weights, we update feature $\mathbf{v}_i \in \mathbb{R}^d$ by fusing the information from its neighborhood, *i.e.,*

$$\mathbf{v}_i = \mathbf{W}_2\, \tilde{\mathbf{v}}_i + \sigma\left(\sum\nolimits_k \widehat{a}_k\, \tilde{\mathbf{v}}_i^k\right), \tag{3}$$

where $\mathbf{W}_2$ is a trainable parameter and σ is the Sigmoid function.

Based on the original relation feature $\bar{\mathbf{r}}_{ij}$, we apply a linear transformation to obtain the updated relation embedding $r_{ij} \in \mathbb{R}^d$. After updating the node and relation embeddings, we need to determine whether there is a relationship between two given entities. An intuitive way is to calculate the probability for each triple. Following ConvKB (Nguyen et al. 2018), we train a scoring function to perform the relation prediction as follows:

$$s_m = \mathrm{FC}([\mathbf{v}_i, \mathbf{r}_m, \mathbf{v}_j] * \mathbf{\Omega}), \tag{4}$$

where $*$ denotes a convolution operation, $\mathbf{\Omega} \in \mathbb{R}^{1\times 3}$ is a set of convolution filters, and $\mathrm{FC}(\cdot)$ is a linear transformation layer. Following (Nathani et al. 2019), we assign a score s_m to the triplet $(\mathbf{v}_i, \mathbf{r}_m, \mathbf{v}_j)$ in Eqn.(4), which indicates the probability that the triplet holds. For each entity, we first traverse all entities and relationships to construct triples, and then we select the triplet with the highest score as the new link. In this way, the *Reader* can capture potential relations between different nodes and derive a new graph $\mathcal{G}_P$. Finally, we denote the enriched knowledge graph as $\mathcal{G} = \mathcal{G}_I \cup \mathcal{G}_P$.

3.2 Writer: Graph-to-Text

Based on the enriched graph $\mathcal{G}$ with N entities, we propose a *Writer* to generate novel paragraphs, which consists of a graph encoder and a text decoder. Specifically, the *Writer* first uses the graph encoder to extract the knowledge representations and then writes a new paragraph with the text decoder (Vaswani et al. 2017).

Graph encoder. A comprehensive understanding of a KG $\mathcal{G}$ is the first step to generate the desired paragraph. However, it is difficult to directly capture rich semantic information in the knowledge graph $\mathcal{G}$. To address this, we extract the knowledge representations within two sub-encoders, *i.e.,* global-graph encoder and local-graph encoder. Following CGE-LW (Ribeiro et al. 2020), we integrate global context information and local topology information to generate new paragraphs.

To aggregate global context information, we first concatenate all of the node features $\mathbf{v}$ and feed them into the global-graph encoder Ψ as follows:

$$[\widehat{\mathbf{v}}_1, \ldots, \widehat{\mathbf{v}}_N] = \Psi([\mathbf{v}_1, \ldots, \mathbf{v}_N]), \tag{5}$$

where Ψ is a standard Transformer encoder (Vaswani et al. 2017), which contains multi-head self-attention layers and feed-forward networks. In the global-graph encoder, we treat the knowledge graphs $\mathcal{G}$ as a fully connected graph without labeled edges. Based on the self-attention mechanism, the global-graph encoder is suitable for discovering the global correlation between nodes. Each node $\widehat{\mathbf{v}}_i \in \mathbb{R}^d$ has the ability to capture all nodes' information.

To better represent the interaction between nodes, we need to build local relations between each node and its neighborhood. However, the global-graph encoder does not explicitly consider such graph topology information. To address this, we use the local-graph encoder to model the local relations. For each node, we first calculate attention weights for its adjacent nodes since the different types of relationships have considerable discrepancies in impact when fusing information. Based on the attention weights, we obtain the hidden node features $\widehat{\mathbf{h}}$ by

$$\begin{aligned} \widehat{\mathbf{h}}_i &= \sum_{j \in \mathcal{N}_i} a_{ij}\, \mathbf{r}_{ij}\, \widehat{\mathbf{v}}_j, \quad \text{where} \\ \widehat{\mathbf{r}}_{ij} &= \mathrm{ReLU}(\mathbf{r}_{ij}\, \mathbf{W}_3\, [\widehat{\mathbf{v}}_i, \widehat{\mathbf{v}}_j]), \\ a_{ij} &= \frac{\exp(\widehat{\mathbf{r}}_{ij})}{\sum_{j \in \mathcal{N}_i} \exp(\widehat{\mathbf{r}}_{ij})}. \end{aligned} \tag{6}$$

Here, $\mathbf{W}_3$ denotes the model parameters and $\widehat{\mathbf{h}}_i$ denotes the hidden features which encode the local interaction between the i-th node and its neighborhood. $\mathcal{N}_i$ denotes the neighbourhood of the i-th node. We also perform the multi-head attention operation to learn structural information from different perspectives. we employ a GRU (Cho et al. 2014) to merge local information between different layers as follows:

$$\mathbf{h}_i = \mathrm{GRU}(\widehat{\mathbf{v}}_i, \widehat{\mathbf{h}}_i), \tag{7}$$

where the final node representation $\mathbf{h}_i \in \mathbb{R}^d$.

Text decoder. Based on the node representations $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^N$, we use the standard Transformer decoder (Vaswani et al. 2017) to generate a novel paragraph τ with T words in an auto-regression manner. At each step t, the text decoder consumes the previously generated tokens as additional input and outputs a probability $\mathbf{p}_t$ over candidate vocabularies. We train the *Writer* with supervised learning as follows:

$$\mathcal{L}_{SL} = -\sum_{t=1}^{T} \mathbf{y}_t \log(\mathbf{p}_t), \tag{8}$$

where $\mathbf{y}_t$ is the ground-truth one-hot vector at step t and generates words by selecting the element with the highest score at this step. In practice, the text decoder also can use other sequence generation models, such as LSTM (Hochreiter and Schmidhuber 1997) and so on.

3.3 Reviewer: Feedback Rewards

The encoder-decoder framework has made great progress in many sequence generation tasks, including text summarization and image captioning. Nevertheless, it suffers from some problems. For each training sample, such a framework tends to use only one word as ground-truth at each generation step, even if other candidate words are also reasonable. This leads to a lack of diversity in the generated text. Moreover, the language is so complex that it requires us to evaluate the quality of the generated paragraph from different dimensions, such as grammatical correctness, topic relevance, language and logic coherence, etc. Inspired by the review process of a research paper, we propose a *Reviewer* module to review the generated paragraph from different dimensions. The output of *Reviewer* can be used as an auxiliary training signal to optimize the *Writer*, which is similar to researchers further polishing the paper based on reviews.

Specifically, we design three feedback rewards in the *Reviewer*. First, we use the metric scores of the generated paragraph as a reward to meet the rules of these metrics. Second, we train a Turing-Test discriminator to determine whether the paragraph is generated by an agent or written by a human, which draws on the idea of adversarial training and requires the paragraph to conform to the natural language specification. Third, we design an alignment module to align the generated paragraphs and the corresponding enriched knowledge graphs, which ensures the correctness and completeness of the generated texts. Different from teacher-forcing methods, the *Reviewer* focuses on sentence-level and graph-level alignment. Given a generated paragraph, however, the above evaluation processes are non-differentiable. As discussed in SeqGAN (Yu et al. 2017), the discrete signals are limited in passing the gradient update from the *Reviewer* to the *Writer*. To address this, we denote the outputs of *Reviewer* as rewards R and maximize expectation rewards $\mathbb{E}[\cdot]$ via reinforcement learning. Formally, the goal of *Reviewer* can be represented by $\max\ \mathbb{E}_{P(\tau;\theta)}[R(\tau)]$, where θ denotes the trainable parameters of our model, and τ is the paragraph generated by the *Writer* based on the generation probability P *w.r.t.* θ. Specifically, the reward function is denoted as

$$R(\tau) = R_1 + \lambda_{AR}\, R_2 + \lambda_{MR}\, R_3, \tag{9}$$

where R_1, R_2, and R_3 correspond to the three modules of the *Reviewer*. λ_{AR} and λ_{MR} control the contribution of the corresponding reward. Following policy gradient methods (Williams 1992; Schulman et al. 2017), we can solve the above problem in batch training as follows:

$$\begin{aligned}\mathcal{L}_{RL} &= -\mathbb{E}_{P(\tau;\theta)}[R(\tau)\ \mathrm{log}P\left(\tau;\theta\right)], \\ &\approx -\frac{1}{B}\sum_{b=1}^{B} R(\tau^{(b)})\ \mathrm{log}P\left(\tau^{(b)};\theta\right),\end{aligned} \tag{10}$$

where B is the training batch size. Now, we introduce these reward modules in detail.

Quality reward. Given a generated paragraph τ, we can calculate some quantitative metrics for it, such as BLEU(Papineni et al. 2002), METEOR (Denkowski and Lavie 2014), CIDEr (Vedantam, Zitnick, and Parikh 2015), *etc*. Directly using these metrics as the training reward can boost the sentence generation quality. In this paper, we simply adopt the BLEU score $R_1 = \mathrm{BLEU}(\tau)$ as the reward since the BLEU score is one of the most popular automated and inexpensive metrics. In practice, the BLEU can be replaced with any metric that needs to be optimized.

Adversarial reward. Based on a paragraph τ, this module acts as a discriminator to determine whether τ is manual annotation (Real) or generated by the machine (Fake). Following (Yu et al. 2017), we use Convolutional Neural Network (CNN) to extract text features since it can capture sequence information and has shown exhibited high performance in the complicated sequence classification task (Zhang and LeCun 2015). Specifically, given a generated paragraph τ, we first concatenate the token embedding as the text representation. We then use different numbers of kernels with different window sizes to extract different features over the text representation and produce a new feature map. After applying a max-pooling operation, we perform a fully connected layer with Sigmoid activation to output a probability, which denotes the probability that the input text is real. The calculation can be formulated as $R_2 = \mathrm{CNN}(\tau)$. Inspired by adversarial training (Cao et al. 2018), this module aims to minimize the performance gap between humans and the *Writer*.

Alignment reward. A paragraph τ is supposed to align its enriched KG $\mathcal{G}$ since τ is generated by *Writer* according to the $\mathcal{G}$. In this sense, we propose to compute the similarity between τ and $\mathcal{G}$ based on the attention mechanism. Given an abstract τ with T words, we first use Long Short-Term Memory (LSTM) to extract text representation $\mathbf{C} = \{\mathbf{c}_t\}$, where $\mathbf{c}_t \in \mathbb{R}^d$, $t \in \{1, \ldots, T\}$. Following AttnGAN (Xu et al. 2018b), we obtain the hidden representation as follows:

$$\mathbf{q}_t = \mathrm{Softmax}\left(\frac{(\mathbf{W}_Q\mathbf{c}_t)(\mathbf{W}_K\mathbf{H})^\top}{\sqrt{d}}\right)\mathbf{W}_V\mathbf{H}, \tag{11}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are trainable parameters, $\sqrt{d}$ is a scaling factor and $\mathbf{H} \in \mathbb{R}^{d\times N}$ are node features obtained from the *Writer*. With the help of the self-attention mechanism (Vaswani et al. 2017), the hidden feature $\mathbf{q}_t \in \mathbb{R}^d$ not only fuses the text representations but also merges graph information. Then, we calculate the cosine similarity as matching score R_3 as follows:

$$R_3 = \sum_{t=1}^{T} \frac{\mathbf{q}_t^\top\ \mathbf{c}_t}{\|\mathbf{q}_t\|\ \|\mathbf{c}_t\|}. \tag{12}$$

Thus far, we can obtain the rewards R_1, R_2, and R_3 from above the *Reviewer* modules. Finally, to train our DRAW network, we define the overall training loss as follows:

$$\mathcal{L} = \mathcal{L}_{SL} + \lambda_{RL}\ \mathcal{L}_{RL}, \tag{13}$$

where λ_{RL} is a trade-off parameter. $\mathcal{L}_{SL}$ trains the DRAW network within supervised learning while $\mathcal{L}_{RL}$ allows the DRAW network to explore diverse generation via reinforcement learning and evaluate the generation from multiple orientations.

4 Experiments

4.1 Datasets

AGENDA dataset. AGENDA is one of the most popular KGs-to-text datasets, which concludes 40,000 pair samples collected from the proceedings of 12 top AI conferences. Each sample consists of a title, an abstract, and the corresponding KG, which is extracted by the SciIE system. The KG is composed of recognized scientific terms and their relationships. In particular, the types of scientific terms include Task, Metric, Method, Material, and Other. The types of relationships include Used-for, Conjunction, Feature-of, Part-of, Compare, Evaluate-for, and Hyponym-of.

M-AGENDA dataset. To further demonstrate the effectiveness of our DRAW network, we create a new dataset, called M-AGENDA. Specifically, we first calculate the cosine similarity between each abstract and the others in the AGENDA dataset. We select two most-related instances for each one and combine these three as a new data example in the M-AGENDA dataset.

4.2 Experimental Settings

Implementation details. Our DRAW network consists of three well-design modules, *i.e., Reader*, *Writer* and *Reviewer*. We first train our *Reader*, *Writer* and *Reviewer* on AGENDA dataset. Then, we use the trained *Reader* and *Writer* model on the M-AGENDA to generate novel paragraphs. To speed up convergence early in training, we adopt different pretraining strategies for each module. For the *Reader*, we first use TransE (Bordes et al. 2013) to train entity and relation embeddings. We then aggregate information passed from a 2-hop neighborhood to update the embedding of each node. Following (Nathani et al. 2019), we use Adam optimization with an initial learning rate of 0.1. For the *Writer*, we pre-train for 30 epochs with early stopping. Following (Ribeiro et al. 2020), we use Adam optimization with an initial learning rate of 0.5. To ensure the generation effect, we set the maximum generation length to 430. For the *Reviewer*, we pre-train the adversarial module with SGD optimization and initialize a learning rate of 0.001. When pre-training the graph encoder of the alignment module, we use the same model and parameters of *writer*. In addition, we systematically adjust the values of λ_{AR} and λ_{MR} to conduct several ablation studies. We find that the experimental results of different coefficient combinations fluctuate only around 0.1, causing little effect on the results. *Writer-Reviewer* obtains the best results with $\lambda_{AR} = \lambda_{MR} = 2$. We set the trade-off parameter $\lambda_{RL} = 1$. We implement our method with PyTorch.

Model	BLEU	METEOR	CIDEr
GraphWriter	14.44	18.80	28.30
GraphWriter+RBS	15.17	19.59	-
Graformer	17.33	21.43	-
CGE-LW	18.01	22.34	33.06
Writer-Reviewer (**Ours**)	**19.60**	**24.03**	**45.21**

Table 1: Quantitative evaluations of generation systems on the AGENDA dataset (higher is better).

Paragraph	Turing Test Results	
	Human	Machine
Written by Human	68%	32%
Written by **DRAW**	48%	52%

Table 2: Quantitative results of Turing test.

Evaluation metrics. To demonstrate the quality of the generated paragraphs, we report both quantitative results and human study results. We divide our evaluation into two parts: KGs-to-text evaluation and overall performance evaluation.

For KGs-to-text evaluations, we adopt three general quantitative evaluation metrics, *i.e.,* BLEU (Papineni et al. 2002), METEOR (Denkowski and Lavie 2014) and CIDEr (Vedantam, Zitnick, and Parikh 2015) to evaluate our *Writer-Reviewer*. In addition, to demonstrate the realness of the paragraphs generated by our model, we also set up a Turing test. Specifically, we randomly select 100 abstracts and shuffle them to find an evaluation set, where half of the abstracts are written by authors and the rest are generated by our *Writer-Reviewer*. After that, we test the turkers on Amazon Mechanical Turk (AMT) to determine whether the paragraphs in the evaluation set are written by humans.

For overall performance evaluation, we set up a human study to rate the abstracts generated by DRAW network, CGE-LW and PaperRobot. For each model, we randomly select 50 generated paragraphs and score them in terms of ‘grammar’, ‘informativeness’, and ‘coherence’ on Amazon Mechanical Turk (AMT). Specifically, the metric ‘grammar’ measures the paragraphs written in well-formed English. The metric ‘informativeness’ denotes whether the paragraphs make use of appropriate scientific terms. The metric ‘coherence’ denotes that the generated text conforms to general specifications. For example, a complete abstract should include a brief introduction to a task, describe the solution, analyze and discuss the results, and so on. Each metric described above, contains 10 levels, with rankings from 1 to 10 (from bad to good).

Following the relation prediction task (Nathani et al. 2019), we evaluate our link prediction method of *Reader* on the proportion of correct entities in the top N ranks (Hits@N) for N=1,3, and 10.

4.3 KGs-to-text Evaluation on AGENDA Dataset

To verify our model on KGs-to-text task, we compare our *Writer-Reviewer* against several state-of-the-art models including GraphWriter (Koncel-Kedziorski et al. 2019), GraphWriter+RBS (An 2019), Graformer (Schmitt et al. 2020) and CGE-LW (Ribeiro et al. 2020) on the AGENDA dataset.

Model	BLEU	METEOR	CIDEr
Writer	18.01	22.34	33.06
Writer+Adversarial	19.37	23.87	39.30
Writer+Alignment	19.33	24.00	43.49
Writer+Quality	19.50	24.03	44.40
Writer-Reviewer (**Ours**)	**19.60**	**24.03**	**45.21**

Table 3: Ablation study for modules used in the *Reviewer* on the AGENDA dataset.

Model	Grammar	Coherence	Informativeness
PaperRobot	5.11	4.95	5.01
CGE-LW	6.77	6.29	6.57
DRAW (Ours)	**7.63**	**6.83**	**7.10**

Table 4: Automatic evaluations results (higher is better).

Results. We report the results of our method and other compared models with respect to three quantitative evaluation metrics in Table 1. As shown in Table 1, our *Writer-Reviewer* achieves better performance than all the compared models in three quantitative evaluation metrics. Specifically, our *Writer-Reviewer* outperforms the state-of-the-art method CGE-LW by 1.6 points in BLEU, 1.7 points in METEOR and 12.2 points in CIDEr. These results demonstrate the superiority of our *Writer-Reviewer* in the KGs-to-text task.

In addition, we carry out a human evaluation to demonstrate the effectiveness of our *Writer-Reviewer*. To be specific, for each paragraph in the evaluation set, we ask the human to choose whether these paragraphs are written by human-authors. From these results in Table 2, nearly half of the paragraphs generated by our *Writer-Reviewer* are reviewed as written by humans. More critically, 32% of the paragraphs written by humans are chosen as written by the AI system. These results demonstrate that our *Writer-Reviewer* can generate realistic paragraphs similar to those written by humans.

Ablation studies in *Reviewer*. To investigate the effect of different modules in *Reviewer*, we conduct an ablation study. As shown in Table 3, *Writer* combined with one of the modules in *Reviewer* arbitrarily obtains better performance than *Writer*, which demonstrates the effectiveness of the modules in *Reviewer*. *Writer* combined with all the modules in *Reviewer*, namely *Writer-Reviewer*, achieves best performance.

4.4 Evaluation on M-AGENDA Dataset

To show the effectiveness of our DRAW network, we conduct experiments on the M-AGENDA dataset. Since the M-AGENDA dataset does not provide ground-truth, we conduct human study instead of quantitative evaluations. Specifically, for each metric in the human study, we average the scores of the paragraphs rated by the humans as the final score.

Results of DRAW. We report the experimental results of our DRAW network and other compared methods in Table 4. From these results, our DRAW network achieves the best performance in terms of ‘grammar’, ‘coherence’, and ‘informativeness’. Specifically, PaperRobot (Wang et al. 2019)

Initial KGs	**41** entities, **18** relations: (global scene-level contextual information, PART-OF, spatial context recurrent convnet model) ; (wikipedia, USED-FOR, multilingual ner systems) ; (local image de-scriptors, CONJUNCTION, spatial configurations) . . .
PaperRobot	In this paper we propose a novel approach for multilingual orangenamed entity recognition tasks . The proposed method is based on semantic similarity measure that can be used to improve word retrieval performance by using wikipedia type of words from text documents and then build an efficient query language model which allows users with similar information between entities as clusters across different domains : part-of-speech tags are generated through each user 's document representation ; our knowledge base system was evaluated over state-of-the-art approaches trained object . . . [*covering 6 entities.*]
CGE-LW	in this paper ,. . . we propose a spatial context recurrent convnet model to incorporate global scene-level contextual information into a spatial context recurrent convnet model for object retrieval .. . . . , and the contextual information from candidate boxes is used for object retrieval. a positional language model that captures contextual information from candidate boxes for object retrieval. the proposed system is evaluated on the tac-kbp 2010 data, and the experimental results show that the proposed system can significantly improve the entity linking performance. . . [*covering 21 entities.*]
DRAW	in this paper , we propose a novel approach to entity linking^1 based on statistical language model-based information retrieval^1 , which exploits both local contexts and global world knowledge^2 to improve the entity linking^2 performance.. . . , we propose a spatial context recurrent convnet model to integrate global context features with local image de-scriptors^3 , spatial configurations , and global scene-level contextual information^3 into a spatial context recurrent convnet model. . . , and a recurrent network with local and global information to guide the search for candidate boxes for object retrieval. . . [*covering 26 entities.*]

Table 5: Example outputs of various models. To better visualize the generated text, we omit information irrelevant to the comparisons. The potential knowledge is represented with the corresponding superscript.

Method	Hits@N		
	@1	@3	@10
PaperRobot	11.9	19.5	42.4
Our	**36.8**	**46.0**	**56.1**

Table 6: Accuracy of the link prediction on the M-AGENDA dataset. Hits@N values are in percentage.

obtains poor performance due to the neglect of the topological structure between entities. CGE-LW (Ribeiro et al. 2020) takes advantage of the graph information effectively and achieves 6.77, 6.29, and 6.57 points in terms of three metrics, but it also ignores the fact that the generated paragraphs are supposed to match the KGs. Different from the methods above, our DRAW network not only performs link prediction with multi-hop information in the *Reader* but also matches the graphs and the generated paragraphs, and thus achieves the best performance. More ablation experiments about *Reader* can be found in the supplementary material.

Results of *Reader*. As shown in Table 6, we report the experimental results of the link prediction method of our *Reader* and PaperRobot. Our method achieves the Hits@1, Hits@3, Hits@10 scores of 36.8, 46.0, and 56.1, outperforming the PaperRobot by 24.5, 26.5, and 13.9 points, respectively. It demonstrates the effectiveness of our link prediction method.

Visualization analysis. As shown in Table 5, we visualize a generated paragraph of our DRAW network. More visualization results can be found in the supplementary material. We see that our DRAW network has the ability to cover more entities , while PaperRobot mentions less entities in the given KG. In addition, CGE-LW tends to repeat unrelated entities/sentences. With the help of *Reviewer*, the generated text of DRAW network is fluent and grammatically correct. Moreover, our DRAW network is able to discover the potential relationships between entities (represented in superscript.)

5 Conclusions and Future Work

In this paper, we propose a Deep ReAder-Writer (DRAW) network that reads multiple AI-related abstracts and then writes a new paragraph to represent enriched knowledge combining the potential knowledge covering the topics mentioned in the source abstracts. Inspired by the review process, we propose a *Reviewer* to rate the quality of the generated texts from different dimensions, which serve as feedback signals to refine our DRAW network. Ablation experiments demonstrate the effectiveness of our method. Moreover, *Writer-Reviewer* achieves state-of-the-art results on KGs-to-text generation task. In terms of human study, some generations of our DRAW network successfully pass the Turing test and confuse the turkers. In future study, we will extend the DRAW network to write a complete paper in an iterative manner and develop more techniques to discover novel ideas, such as creating new entities.

Acknowledgments

This work was partially supported by Key-Area Research and Development Program of Guangdong Province

2018B010107001, National Natural Science Foundation of China (NSFC) 61836003 (key project), Program for Guangdong Introducing Innovative and Entrepreneurial Teams 2017ZT07X183, International Cooperation open Project of State Key Laboratory of Subtropical Building Science, South China University of Technology (2019ZA01), Fundamental Research Funds for the Central Universities D2191240.

References

An, B. 2019. Repulsive Bayesian Sampling for Diversified Attention Modeling. In *workshop of NeurIPS*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NeurIPS*.

Buenz, E. J. 2019. Essential elements for high-impact scientific writing. *Nature* doi: 10.1038/d41586-019-00546-7.

Cao, J.; Guo, Y.; Wu, Q.; Shen, C.; Huang, J.; and Tan, M. 2018. Adversarial Learning with Local Coordinate Coding. In *ICML*.

Cao, J.; Guo, Y.; Wu, Q.; Shen, C.; Huang, J.; and Tan, M. 2020. Improving Generative Adversarial Networks with Local Coordinate Coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence* .

Cao, J.; Mo, L.; Zhang, Y.; Jia, K.; Shen, C.; and Tan, M. 2019. Multi-marginal wasserstein gan. In *NeurIPS*.

Chen, P.; Zhang, Y.; Tan, M.; Xiao, H.; Huang, D.; and Gan, C. 2020. Generating Visually Aligned Sound From Videos. *IEEE Trans. Image Process.* 29: 8292–8302.

Cho, K.; Merrienboer, B. V.; Çaglar Gülçehre; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

Denkowski, M. J.; and Lavie, A. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *ACL*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. In *AAAI*.

Feng, N.; Jinpeng, W.; Jin-Ge, Y.; Rong, P.; and Chin-Yew, L. 2018. Operation-guided Neural Networks for High Fidelity Data-To-Text Generation. In *EMNLP*.

Gerber, M.; and Chai, J. 2010. Beyond NomBank: A Study of Implicit Arguments for Nominal Predicates. In *ACL*.

Gopen, G. D.; and Ja, S. 1990. The Science of Scientific Writing. *American Scientist* .

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-term Memory. *Neural Computation* .

Huang, D.; Chen, P.; Zeng, R.; Du, Q.; Tan, M.; and Gan, C. 2020. Location-Aware Graph Convolutional Networks for Video Question Answering. In *AAAI*, 11021–11028.

Kipf, T.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Koncel-Kedziorski, R.; Bekal, D.; Luan, Y.; Lapata, M.; and Hajishirzi, H. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *NAACL-HLT*.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*.

Ling, F. U.; and Hui, Z. 2013. Multi-document summary using LDA and spectral clustering. *Computer Engineering & Applications* .

Luan, Y.; He, L.; Ostendorf, M.; and Hajishirzi, H. 2018. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *EMNLP*.

Min, Z.; Jie, Z.; Jian, S.; and GuoDong, Z. 2006. A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features. In *ACL*.

Nathani, D.; Chauhan, J.; Sharma, C.; and Kaul, M. 2019. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In *ACL*.

Nguyen, D. Q.; Nguyen, T.; Nguyen, D. Q.; and Phung, D. Q. 2018. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *NAACL-HLT*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *ACL*.

Ribeiro, L.; Gardent, C.; and Gurevych, I. 2019. Enhancing AMR-to-Text Generation with Dual Graph Representations. In *EMNLP-IJCNLP*.

Ribeiro, L. F. R.; Zhang, Y.; Gardent, C.; and Gurevych, I. 2020. Modeling Global and Local Node Contexts for Text Generation from Knowledge Graphs. *Transactions of the Association for Computational Linguistics* .

Schlichtkrull, M.; Kipf, T.; Bloem, P.; Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*.

Schmitt, M.; Ribeiro, L. F. R.; Dufter, P.; Gurevych, I.; and Schutze, H. 2020. Modeling Graph Structure via Relative Position for Better Text Generation from Knowledge Graphs. *ArXiv* abs/2006.09242.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *ArXiv* abs/1707.06347.

Trisedya, B.; Jianzhong, Q.; Rui, Z.; and Wei, W. 2018. GTR-LSTM: A Triple Encoder for Sentence Generation from RDF Data. In *ACL*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *NeurIPS*.

Vedantam, R.; Zitnick, C. L.; and Parikh, D. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR*.

Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Xie, X.; and Guo, M. 2018. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*.

Wang, Q.; Huang, L.; Jiang, Z.; Knight, K.; Ji, H.; Bansal, M.; and Luan, Y. 2019. PaperRobot: Incremental Draft Generation of Scientific Ideas. In *ACL*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* .

Xiao, L.; Wang, L.; He, H.; and Jin, Y. 2020. Copy or Rewrite: Hybrid Summarization with Hierarchical Reinforcement Learning. In *AAAI*.

Xu, K.; Wu, L.; guo Wang, Z.; Yu, M.; Chen, L.; and Sheinin, V. 2018a. SQL-to-Text Generation with Graph-to-Sequence Model. In *EMNLP*.

Xu, T.; Zhang, P.; Huang, Q.; Zhang, H.; Gan, Z.; Huang, X.; and He, X. 2018b. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks. In *CVPR*.

Yoshikawa, K.; Riedel, S.; Hirao, T.; Asahara, M.; and Matsumoto, Y. 2010. Coreference based event-argument relation extraction on biomedical text. *Journal of Biomedical Semantics* .

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*.

Zhang, X.; and LeCun, Y. 2015. Text Understanding from Scratch. *ArXiv* abs/1502.01710.

Zhen, W.; Jianwen, Z.; Jianlin, F.; and Zheng, C. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*.

Zhijiang, G.; Yan, Z.; Zhiyang, T.; and Wei, L. 2019. Densely Connected Graph Convolutional Networks for Graph-to-Sequence Learning. *Transactions of the Association for Computational Linguistics* .