Text Generation from Knowledge Graphs with Graph Transformers

Rik Koncel-Kedziorski[1], Dhanush Bekal[1], Yi Luan[1], Mirella Lapata[2], and Hannaneh Hajishirzi[1,3]

[1]University of Washington
{kedzior,dhanush,luanyi,hannaneh}@uw.edu
[2]University of Edinburgh
mlap@inf.ed.ac.uk
[3]Allen Institute for Artificial Intelligence

Abstract

Generating texts which express complex ideas spanning multiple sentences requires a structured representation of their content (document plan), but these representations are prohibitively expensive to manually produce. In this work, we address the problem of generating coherent multi-sentence texts from the output of an information extraction system, and in particular a knowledge graph. Graphical knowledge representations are ubiquitous in computing, but pose a significant challenge for text generation techniques due to their non-hierarchical nature, collapsing of long-distance dependencies, and structural variety. We introduce a novel graph transforming encoder which can leverage the relational structure of such knowledge graphs without imposing linearization or hierarchical constraints. Incorporated into an encoder-decoder setup, we provide an end-to-end trainable system for graph-to-text generation that we apply to the domain of scientific text. Automatic and human evaluations show that our technique produces more informative texts which exhibit better document structure than competitive encoder-decoder methods. [1]

1 Introduction

Increases in computing power and model capacity have made it possible to generate mostly-grammatical sentence-length strings of natural language text. However, generating several sentences related to a topic and which display overall coherence and discourse-relatedness is an open challenge. The difficulties are compounded in domains of interest such as scientific writing. Here the variety of possible topics is great (e.g. topics as diverse as driving, writing poetry, and picking stocks are all referenced in one subfield of one scientific discipline). Additionally, there are strong constraints on document structure, as scientific communication requires carefully ordered explanations of processes and phenomena.

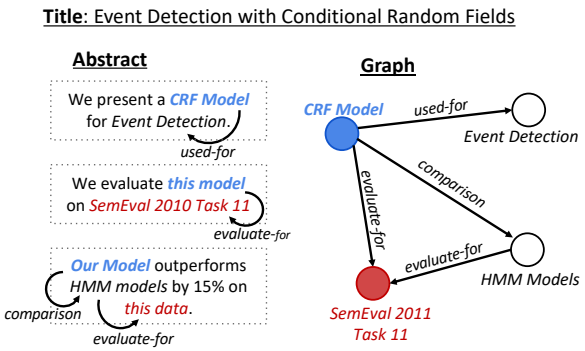

Figure 1: A scientific text showing the annotations of an information extraction system and the corresponding graphical representation. Coreference annotations shown in color. Our model learns to generate texts from automatically extracted knowledge using a graph encoder decoder setup.

Many researchers have sought to address these issues by working with structured inputs. Data-to-text generation models (Konstas and Lapata, 2013; Lebret et al., 2016; Wiseman et al., 2017; Puduppully et al., 2019) condition text generation on table-structured inputs. Tabular input representations provide more guidance for producing longer texts, but are only available for limited domains as they are assembled at great expense by manual annotation processes.

The current work explores the possibility of using information extraction (IE) systems to automatically provide context for generating longer texts (Figure 1). Robust IE systems are available and have support over a large variety of textual domains, and often provide rich annotations of relationships that extend beyond the scope of

[1]Data and code available at https://github.com/rikdz/GraphWriter

Proceedings of NAACL-HLT 2019, pages 2284–2293
Minneapolis, Minnesota, June 2 - June 7, 2019.

a single sentence. But due to their automatic nature, they also introduce challenges for generation such as erroneous annotations, structural variety, and significant abstraction of surface textual features (such as grammatical relations or predicate-argument structure).

To effect our study, we use a collection of abstracts from a corpus of scientific articles (Ammar et al., 2018). We extract entity, coreference, and relation annotations for each abstract with a state-of-the-art information extraction system (Luan et al., 2018), and represent the annotations as a *knowledge graph* which collapses co-referential entities. An example of a text and graph are shown in Figure 1. We use these graph/text pairs to train a novel attention-based encoder-decoder model for knowledge-graph-to-text generation. Our model, GraphWriter, extends the successful Transformer for text encoding (Vaswani et al., 2017) to graph-structured inputs, building on the recent Graph Attention Network architecture (Veličković et al., 2018). The result is a powerful, general model for graph encoding which can incorporate global structural information when contextualizing vertices in their local neighborhoods.

The main contributions of this work include:

1. We propose a new graph transformer encoder that applies the successful sequence transformer to graph structured inputs.
2. We show how IE output can be formed as a connected unlabeled graph for use in attention-based encoders.
3. We provide a large dataset of knowledge-graphs paired with scientific texts for further study.

Through detailed automatic and human evaluations, we demonstrate that automatically extracted knowledge can be used for multi-sentence text generation. We further show that structuring and encoding this knowledge as a graph leads to improved generation performance compared to other encoder-decoder setups. Finally, we show that GraphWriter's transformer-style encoder is more effective than Graph Attention Networks on the knowledge-graph-to-text task.

2 Related Work

Our work falls under the larger scope of concept-to-text generation. Barzilay and Lapata (2005) introduced a collective content selection model for generating summaries of football games from tables of game statistics. Liang et al. (2009) jointly learn to segment and align text with records, reducing the supervision needed for learning. Kim and Mooney (2010) improve this technique by learning a semantic parse to logical forms. Konstas and Lapata (2013) focus on the generation objective, jointly learning planning and generating using a rhetorical (RST) grammar induction approach.

These earlier works often focused on smaller record generation datasets such as WeatherGov and RoboCup, but recently Mei et al. (2016) showed how neural models can achieve strong results on these standards, prompting researchers to investigate more challenging domains such as ours.

Lebret et al. (2016) tackles the task of generating the first sentence of a Wikipedia entry from the associated infobox. They provide a large dataset of such entries and a language model conditioned on tables. Our work focuses on a multi-sentence task where relations can extend beyond sentence boundaries.

Wiseman et al. (2017) study the difficulty of applying neural models to the data-to-text task. They introduce a large dataset where a text summary of a basketball game is paired with two tables of relevant statistics and show that neural models struggle to compete with template based methods over this data. We propose generating from graphs rather than tables, and show that graphs can be effectively encoded to capture both local and global structure in the input.

We show that modeling knowledge as a graph improves generation results, connecting our work to other graph-to-text tasks such as generating from Abstract Meaning Representation (AMR) graphs. Konstas et al. (2017) provide the first neural model for this task, and show that pretraining on a large dataset of noisy automatic parses can improve results. However, they do not directly model the graph structure, relying on linearization and sequence encoding instead. Current works improve this through more sophisticated graph encoding techniques. Marcheggiani and Perez-Beltrachini (2018) encode input graphs directly using a graph convolution encoder (Kipf and Welling, 2017). Our model extends the graph attention networks of Veličković et al. (2018), a direct descendant of the convolutional approach which offers more modeling power and has been

	Title	Abstract	KG
Vocab	29K	77K	54K
Tokens	413K	5.8M	1.2M
Entities	-	-	518K
Avg Length	9.9	141.2	-
Avg #Vertices	-	-	12.42
Avg #Edges	-	-	4.43

Table 1: Data statistics of our AGENDA dataset. Averages are computed per instance.

shown to improve performance. Song et al. (2018) uses a graph LSTM model to effect information propagation. At each timestep, a vertex is represented by a gated combination of the vertices to which it is connected and the labeled edges connecting them. Beck et al. (2018) use a similar gated graph neural network. Both of these gated models make heavy use of label information, which is much sparser in our knowledge graphs than in AMR. Generally, AMR graphs are denser, rooted, and connected, whereas the knowledge our model works with lacks these characteristics. For this reason, we focus on attention-based models such as Veličković et al. (2018), which impose fewer constraints on their input.

Finally, our work is related to Wang et al. (2018) who offer a method for generating scientific abstracts from titles. Their model uses a gated rewriter network to write and revise several draft outputs in several sequence-to-sequence steps. While we operate in the same general domain as this work, our task setup is ultimately different due to the use of extracted information as input. We argue that our setup improves the task defined in Wang et al. (2018), and our more general model can be applied across tasks and domains.

3 The AGENDA Dataset

We consider the problem of generating a text from automatically extracted information (*knowledge*). IE systems can produce high quality knowledge for a variety of domains, synthesizing information from across sentence and even document boundaries. Generating coherent text from knowledge requires a model which considers global characteristics of the knowledge as well as local characteristics of each entity. This feature of the task motivates our use of graphs for representing knowledge, where neighborhoods localize important information and paths through the graph build connections between distant nodes through intermediate ones. An example knowledge graph can be seen in Figure 1.

We formulate our problem as follows: given the title of a scientific article and a knowledge graph constructed by an automatic information extraction system, the goal is to generate an abstract that a) is appropriate for the given title and b) expresses the content of the knowledge graph in natural language text. To evaluate how well a model accomplishes this goal, we introduce the Abstract GENeration DAtaset (AGENDA), a dataset of knowledge graphs paired with scientific abstracts. Our dataset consists of 40k paper titles and abstracts from the Semantic Scholar Corpus taken from the proceedings of 12 top AI conferences (Ammar et al., 2018).

For each abstract, we create a knowledge graph in two steps. First, we apply the SciIE system of Luan et al. (2018), a state-of-the-art science-domain information extraction system. This system provides named entity recognition for scientific terms, with entity types Task, Method, Metric, Material, or Other Scientific Term. The model also produces co-reference annotations as well as seven relations that can obtain between different entities (Compare, Used-for, Feature-of, Hyponym-of, Evaluate-for, and Conjunction). For example, in Figure 1, the node labeled "SemEval 2011 Task 11" is of type 'Task', "HMM Models" is of type 'Model', and there is a 'Evaluate-For' relation showing that the models are evaluated on the task.

We form these annotations into knowledge graphs. We collapse co-referential entities into a single node associated with the longest mention (on the assumption that these will be the most informative). We then connect nodes to one another using the relation annotations, treating these as labeled edges in the graph. The result is a possibly unconnected graph representation of the SciIE annotations for a given abstract.

Statistics of the AGENDA dataset are available in Table 1. We split the AGENDA dataset into 38,720 training, 1000 validation, and 1000 test datapoints. We offer standardized data splits to facilitate comparison.

4 Model

Following most work on neural generation we adopt an encoder-decoder architecture, shown in

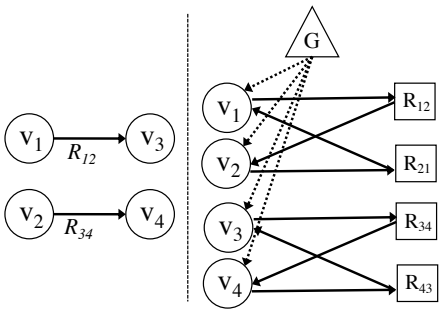

Figure 2: Converting disconnected labeled graph to connected unlabeled graph for use in attention-based encoder. v_i refer to vertices, R_{ij} to relations, and G is a global context node.

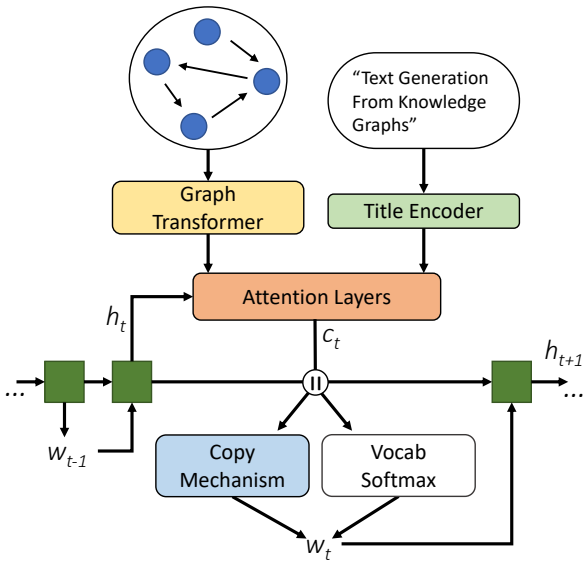

Figure 3: GraphWriter Model Overview

Figure 3, which we call GraphWriter. The input to GraphWriter is a title and a knowledge graph which are encoded respectively with a bidirectional recurrent neural network and a novel Graph Transformer architecture (to be discussed in Section 4.1). At each decoder time step, we attend on encodings of the knowledge graph and document title using the decoder hidden state $\mathbf{h}_t \in \mathbb{R}^d$. The resulting vectors are used to select output w_t either from the decoder's vocabulary or by copying an entity from the knowledge graph. Details of our decoding process are described in Section 4.2. The model is trained end-to-end to minimize the negative log likelihood of the mixed copy and vocabulary probability distribution and the human authored text.

4.1 Encoder

The AGENDA dataset contains a knowledge graph for each datapoint, but our model requires unlabeled, connected graphs as input. To encode knowledge graphs with this model, we restructure each graph as an unlabeled connected graph, preserving label information by the method described below and sketched in Figure 2.

Graph Preparation We convert each graph to an unlabeled connected bipartite graphs following a similar procedure to Beck et al. (2018). In this process, each labeled edge is replaced with two vertices: one representing the forward direction of the relation and one representing the reverse. These new vertices are then connected to the entity vertices so that the directionality of the former edge is maintained. This restructures the original knowledge graph as an unlabeled directed graph where all vertices correspond to entities and relations in the SciIE annotations without loss of information. To promote information flow between disconnected parts of the graph, we add a global vertex which connects all entity vertices. This global vertex will be used to initialize the decoder, analogously to the final encoder hidden state in a traditional sequence to sequence model. The final result of these restructuring operations is a connected, unlabeled graph $G = (V, E)$, where V is a list of entities, relations, and a global node and E is an adjacency matrix describing the directed edges.

Graph Transformer Our model is most similar to the Graph Attention Network (GAT) of Veličković et al. (2018), which computes the hidden representations of each node in a graph by attending over its neighbors following a self-attention strategy. The use of self-attention in GAT addresses the shortcomings of prior methods based on graph convolutions (Defferrard et al., 2016; Kipf and Welling, 2017), but limits vertex updates to information from adjacent nodes. Our model allows for a more global contextualization of each vertex through the use of a transformer-style architecture. The recently proposed Transformer (Vaswani et al., 2017) addresses the inherent sequential computation shortcoming of recurrent neural networks, enabling efficient and paralleled computation by invoking a self-attention mechanism for global context modeling. These models have shown promising results in a variety of text processing tasks (Radford et al., 2018).

Our Graph Transformer encoder starts with self-

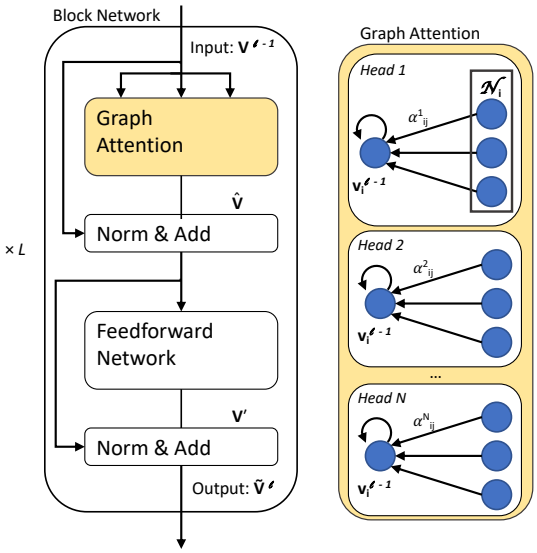

Figure 4: Graph Transformer

attention of local neighborhoods of vertices; the key difference with GAT is that our model includes additional mechanisms for capturing global context. This additional modeling power allows the Graph Transformer to better articulate how a vertex should be updated given the content of its neighbors, as well as to learn global patterns of graph structure relevant to the model's objective.

Specifically, V is embedded in a dense continuous space by the embedding process described at the end of this section, resulting in matrix $\mathbf{V}^0 = [\mathbf{v}_i], \mathbf{v}_i \in \mathbb{R}^d$ which will serve as input to the graph transformer model shown in Figure 4. Each vertex representation $\mathbf{v}_i$ is contextualized by attending over the other vertices to which v_i is connected in G. We use an N-headed self attention setup, where N independent attentions are calculated and concatenated before a residual connection is applied:

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \Big\|_{n=1}^{N} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^n \mathbf{W}_V^n \mathbf{v}_j \quad (1)$$

$$\alpha_{ij}^n = a^n(\mathbf{v}_i, \mathbf{v}_j) \quad (2)$$

Here, $\|$ denotes the concatenation of the N attention heads, $\mathcal{N}_i$ denotes the neighborhood of v_i in G, $\mathbf{W}_V^n \in \mathbb{R}^{d \times d}$, and where a^n are attention mechanisms parameterized per head. In this work, we use attention functions of the following form:

$$a(\mathbf{q}_i, \mathbf{k}_j) = \frac{\exp((\mathbf{W}_K \mathbf{k}_j)^\top \mathbf{W}_Q \mathbf{q}_i)}{\sum_{z \in \mathcal{N}_i} \exp((\mathbf{W}_K \mathbf{k}_z)^\top \mathbf{W}_Q \mathbf{q}_i)} \quad (3)$$

Each a learns independent transformations $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d}$ of $\mathbf{q}$ and $\mathbf{k}$ respectively, and the resulting product is normalized across all connected edges. To reduce the tendency of these dot products to impede gradient flow, we scale them by $\frac{1}{\sqrt{d}}$, following Vaswani et al. (2017).

The Graph Transformer then augments these multi-headed attention layers with *block* networks. Each block applies the following transformations:

$$\tilde{\mathbf{v}}_i = \text{LayerNorm}(\mathbf{v}'_i + \text{LayerNorm}(\hat{\mathbf{v}}_i)) \quad (4)$$

$$\mathbf{v}'_i = \text{FFN}(\text{LayerNorm}(\hat{\mathbf{v}}_i)) \quad (5)$$

Where $\text{FFN}(\mathbf{x})$ is a two layer feedforward network with a non-linear transformation f between layers i.e. $f(\mathbf{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$.

Stacking multiple blocks allows information to propagate through the graph. Blocks are stacked L times, with the output of layer $l-1$ taken as the input to layer l, so that $\mathbf{v}_i^l = \tilde{\mathbf{v}}_i^{l-1}$. The resulting vertex encodings $\mathbf{V}^L = [\mathbf{v}_i^L]$ represent entities, relations, and the global node contextualized by their relationships in the graph structure. We refer to the resulting encodings as *graph contextualized vertex encodings*.

Embedding Vertices, Encoding Title As stated above, the vertices of our graph correspond to entities and relations from the SciIE annotations. Because each relation is represented as both a forward- and backward-looking vertex, we learn two embeddings per relation as well as an initial embedding for the global node. Entities correspond to scientific terms which are often multi-word expressions. To produce a single d-dimensional embedding per phrase, we use the last hidden state of a bidirectional RNN run over embeddings of each word in the entity phrase, i.e. $\text{BiRNN}(\mathbf{x}_1 \dots \mathbf{x}_m)$ for dense embeddings $\mathbf{x}$ and phrase length m. The output of our embedding step is a collection $\mathbf{V}^0$ of d-dimensional vectors representing each vertex in V.

The title input is also a short string, and so we encode it with another BiRNN to produce $\mathbf{T} = \text{BiRNN}(x'_1 \dots x'_m)$ for title word embedding $\mathbf{x}'$.

4.2 Decoder

We decode with an attention-based decoder with a copy mechanism for copying input from the knowledge graph and title. At each decoding timestep t we use decoder hidden state $\mathbf{h}_t$ to compute context vectors $\mathbf{c}_g$ and $\mathbf{c}_s$ for the graph and

title sequence respectively. $\mathbf{c}_g$ is computed using multi-headed attention contextualized by $\mathbf{h}_t$:

$$\mathbf{c}_g = \mathbf{h}_t + \Big\|_{n=1}^{N} \sum_{j \in V} \alpha_j^n \mathbf{W}_G^n \mathbf{v^L}_j \quad (6)$$

$$\alpha_j = a(\mathbf{h}_t, \mathbf{v^L}_j) \quad (7)$$

for a as described in Equation (1) by attending over the graph contextualized encodings $\mathbf{V}^L$. $\mathbf{c}_s$ is computed similarly, attending over the title encoding $\mathbf{T}$. We then construct the final context vector by concatenation, $\mathbf{c}_t = [\mathbf{c}_g \| \mathbf{c}_s]$. We use an input-feeding decoder (Luong et al., 2015) where both $\mathbf{h}_t$ and $\mathbf{c}_t$ are passed as input to the next RNN timestep.

We compute a probability p of copying from the input using $\mathbf{h}_t$ and $\mathbf{c}_t$ in a fashion similar to See et al. (2017), that is:

$$p = \sigma(\mathbf{W}_{copy}[\mathbf{h}_t \| \mathbf{c}_t] + b_{copy}) \quad (8)$$

The final next-token probability distribution is:

$$p * \alpha^{copy} + (1 - p) * \alpha^{vocab}, \quad (9)$$

Where the probability distribution α^{copy} over entities and input tokens is computed as $\alpha_j^{copy} = a([\mathbf{h}_t \| \mathbf{c}_t], \mathbf{x}_j)$ for $\mathbf{x}_j \in \mathbf{V} \| \mathbf{T}$. The remaining $1-p$ probability is given to α^{vocab}, which is calculated by scaling $[\mathbf{h}_t \| \mathbf{c}_t]$ to the vocabulary size and taking a softmax.

5 Experiments

Evaluation Metrics We evaluate using a combination of human and automatic evaluations. For **human** evaluation, participants were asked to compare abstracts generated by various models and those written by the authors of the scientific articles. We used Best-Worst Scaling (BWS; (Louviere and Woodworth, 1991; Louviere et al., 2015)), a less labor-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2016). Participants were presented with two or three abstracts and asked to decide which one was better and which one was worse in order of grammar and fluency (is the abstract written in well-formed English?), coherence (does the abstract have an introduction, state the problem or task, describe a solution, and discuss evaluations or results?), and informativeness (does the abstract relate to the provided title and make use of appropriate scientific terms?). We provided examples of good and bad abstracts and explain how they succeed or fail to meet the defined criteria.

Because our dataset is scientific in nature, evaluations must be done by experts and we can only collect a limited number of these high quality datapoints.[2] The study was conducted by 15 experts (i.e. computer science students) who were familiar with the abstract writing task and the content of the abstracts they judged. To supplement this, we also provide **automatic** metrics. We use BLEU (Papineni et al., 2002), an n-gram overlap measure popular in text generation tasks, and METEOR (Denkowski and Lavie, 2014), a machine translation with paraphrase and language-specific considerations.

Comparisons We compare our GraphWriter against several strong baselines. In GAT, we replace our Graph Transformer encoder with a Graph Attention Network of (Veličković et al., 2018). This encoder consists of PReLU activations stacked between 6 self-attention layers. To determine the usefulness of including graph relations, we compare to a model which uses only entities and title (EntityWriter). Finally, we compare with the gated rewriter model of Wang et al. (2018) (Rewriter). This model uses only the document title to iteratively rewrite drafts of its output. [3]

Implementation Details Our models are trained end-to-end to minimize the negative joint log likelihood of the target text vocabulary and the copied entity indices. We use SGD optimization with momentum (Qian, 1999) and "warm restarts", a cyclical regiment that reduces the learning rate from 0.25 to 0.05 over the course of 5 epochs, then resets for the following epoch. Models are trained for 15 epochs with early stopping (Prechelt, 1998) based on the validation loss, with most models stopping between 8 and 13 epochs. We use single-layer LSTMs (Hochreiter and Schmidhuber, 1997) as recurrent networks. We use dropout (Srivastava et al., 2014) in self attention layers set to 0.3. Hidden states and embedding dimensions are fixed at 500 and attentions learn 500 dimen-

[2] Attempts to crowd source this evaluation failed.

[3] Due to the larger size and greater variety of our dataset and accompanying vocabularies compared to theirs, we were unable to train this model with the reported batch size of 240. We use batch size 24 instead, which is partially responsible for the lower performance.

	BLEU	METEOR
GraphWriter	**14.3** $\pm$ 1.01	**18.8** $\pm$ 0.28
GAT	12.2 $\pm$ 0.44	17.2 $\pm$ 0.63
EntityWriter	10.38	16.53
Rewriter	1.05	8.38

Table 2: Automatic Evaluations of Generation Systems.

	Best	Worst
Rewriter (No knowledge)	12%	64%
GraphWriter (Knowledge)	24%	36%
Human Authored	64%	0%

Table 3: Does knowledge improve generation? Human evaluations of best and worst abstract.

	Win	Lose	Tie
Structure	63%	17%	20%
Informativeness	43%	23%	33%
Grammar	63%	23%	13%
Overall	63%	17%	20%

Table 4: Human Judgments of GraphWriter and EntityWriter models.

sional projections. In Block layers, the feedforward network has an intermediate size of 2000, and we use a PReLU activation function (He et al., 2015). GraphWriter and GAT use $L = 6$ layers. The number of attention heads is set to 4. In all models, for both inputs and output, we replace words occurring fewer than 5 times with $<unk>$ tokens. In each abstract, we replace all mentions in a coreference chain in the abstract with the canonical mention used in the graph. We decode with beam search (Graves, 2012; Sutskever et al., 2014) with a beam size of 4. A post-processing step deletes repeated sentences and repeated coordinated clauses.

5.1 Results

A comparison of all systems in terms of automatic metrics is shown in Table 2. Our GraphWriter model outperforms other methods. We see that models which leverage title, entities, and relations (GraphWriter and GAT) outperform models which use less information (EntityWriter and Rewriter).

We see that GraphWriter outperforms GAT across metrics, indicating that the global contextualization provided by GraphWriter improves generation. To verify the performance gap between GraphWriter and GAT, we report the average test metrics for 4 training runs of each model along with their variances. We see that the variance of the different models is non-overlapping, and in fact all training runs of GraphWriter outperformed all runs of GAT on these metrics.

Does Knowledge Help? To evaluate the value of knowledge in the generation task we compare our GraphWriter model to a model which does not generate from knowledge. We provide expert annotators with 50 randomly-selected paper titles from the test set and ask them for a single judgment according to the criteria described in Section 5. We pair each paper title with the generated abstracts produced by GraphWriter (a knowledge-informed modes), Rewriter (a knowledge-agnostic model), and the gold abstract (with canonicalized coreferential mentions).

Results of this comparison can be seen in Table 3. We see that GraphWriter is selected as "Best" more often than Rewriter, and is less often selected as "Worst", attesting to the value of including knowledge in the text generation process. We see that sometimes generated texts are preferred to human authored text, which is due in part to the disfluencies introduced by canonicalization of entity mentions.

To further understand the advantages of using knowledge graphs, we provide a more detailed comparison of the GraphWriter and EntityWriter models. We select 30 additional test datapoints and ask experts to provide per-criterion judgments of the outputs of the two systems. Since both models make use of extracted entities, we show this list along with the title for each datapoint, and modify the description of Informativeness to include "making use of the provided entities". Results of this evaluation are shown in Table 4. Here we see that including structured knowledge in the form of a graph improves abstract generation compared to generating from an unstructured collection of entities. The largest gains are made in terms of document structure and grammar, indicating that the structure of the input knowledge is being translated into the surface form.

Generating from Title The Rewriter model (Wang et al., 2018) considers the task of generating an abstract with only the paper's title as input. We compare against this model because it is among the first end-to-end systems to attempt to write scientific abstracts. However, the task setup used in Wang et al. (2018) differs significantly from the task introduced in this work. In order

Title	Block and Group Regularized Sparse Modeling for Dictionary Learning
Knowledge	(dictionary learning, CONJUNCTION, sparse coding) ; (optimization problems, USED-FOR, dictionary learning) ; (optimization problems, USED-FOR, sparse coding)...
GraphWriter	Sparse representations have recently been shown to be effective in many optimization problems. However, existing dictionary learning methods are limited in the number of dictionary blocks, which can be expensive to obtain. In this paper, we propose a novel approach to dictionary learning based on sparse coding ...
GAT	In this paper, we consider the problem of dictionary learning in well-known datasets. In particular, we consider the problem of dictionary learning, where the goal is to find a set of dictionary blocks that maximize the likelihood of a given set of dictionary blocks ...
EntityWriter	We propose a novel dictionary learning framework for reconstructed block/group sparse coding schemes. The dictionary learning framework is based on the descent, which is a block structure of the group structure ...
Rewriter	This paper presents a new approach to the k-means of the algorithm. The proposed approach is based on the basis of the stationarity algorithm. The algorithm is based on the fact that the number of bits is a constant of the base of the base of the input ...
Gold	This paper proposes a dictionary learning framework that combines the proposed block/group (BGSC) or reconstructed block/group (R-BGSC) sparse coding schemes with the novel Intra-block Coherence Suppression Dictionary Learning algorithm. An important and distinguishing feature of the proposed framework is that all dictionary blocks are trained simultaneously ...
Title	Image Interpolation with Directionlets
Knowledge	(directionally adaptive image interpolation USED-FOR edge information) ; (numeric and visual quality, HYPONYM-OF, directionally adaptive image interpolation) ; (directionlets, EVALUATE-FOR, multiple-direction wavelet transform) ...
GraphWriter	In this paper, we propose a novel directionally adaptive image interpolation based on the multiple-direction wavelet transform, called directionlets, which can be used as a directionlets to improve the numeric and visual quality of the directionally adaptive image interpolation ...
GAT	In this paper, we propose a novel directionally adaptive image interpolation, called directionally adaptive image interpolation, for directionally adaptive image interpolation , which is based on the multiple-direction wavelet transform ...
EntityWriter	We present a novel directionally adaptive image interpolation for numeric and visual quality. The wavelet transform is based on the wavelet transform between the low-resolution image and the interpolated image. The high-resolution image is represented by a wavelet transform ...
Rewriter	We present a new method for finding topic-specific data sets. The key technical contributions of our approach is to be a function of the terrestrial distributed memory. The key idea is to be a function of the page that seeks to be ranked the buckets of the data. The basic idea is a new tool for the embedded space ...
Gold	We present a novel directionally adaptive image interpolation based on a multiple-direction wavelet transform, called directionlets. The directionally adaptive image interpolation uses directionlets to efficiently capture directional features and to extract edge information along different directions from the low-resolution image ...

Table 5: Example outputs of various systems versus Gold.

to make a fair comparison, we construct a variant of our model which is only provided with a title as input. We develop a model that predicts entities from the title, and then uses our knowledge-aware model to generate the abstract. For this comparison we use the EntityWriter model with a collection of entities inferred from the title alone (InferEntityWriter).

To infer relevant entities, we learn to embed titles and entities extracted from the corresponding abstract in a shared dense vector space by minimizing their cosine distance. We use negative sampling to provide definition to this vector space. At test time, we use the title embedding to infer the $K = 12$ closest entities to feed into the InferEntityWriter model. Results are shown in Table 6, which shows that InferEntityWriter achieves better results than Rewriter, indicating that the intermediate entity prediction step is helpful in abstract generation.

	BLEU	METEOR
Rewriter	1.05	8.38
InferEntityWriter	3.60	12.2

Table 6: Comparison of generation without knowledge and with Inferred Knowledge (InferEntityWriter)

5.2 Analysis

Table 5 shows examples of various system outputs for a particular test instance.We see that GraphWriter makes use of more entities from the input, arranged with more articulated textual context. It demonstrates less repetition than GAT. Both GraphWriter and GAT show much better coher-

ence than EntityWriter, which copies entities from the input into unreasonable contexts. Rewriter, while fluent and grammatical, jumps from topic to topic, failing to relate as strongly to the input as the knowledge-aware models.

To determine the shortcomings of our model, we calculate rough error statistics over the outputs of the GraphWriter on the test set. We notice that 40% of entities in the knowledge graphs do not appear in the generated text. Future work should address this coverage problem, perhaps through modifications to the inference procedure or a coverage loss (Tu et al., 2016) modified to the specifics of this task. We find that 18% of all sentences generated by our model repeat sentences or clauses and are subjected to the post-processing pruning mentioned in Section 5. While this step is a simple solution to improve generated outputs, a more advanced solution is required.

6 Conclusion

We have studied the problem of generating multi-sentence text from the output of automatic information extraction systems, and have shown that incorporating knowledge as graphs improves performance. We introduced GraphWriter, featuring a new attention model for graph encoding, and demonstrated its utility through human and automatic evaluation compared to strong baselines. Lastly, we provide a new resource for the generation community, the AGENDA dataset of abstracts and knowledge. Future work could address the problem of repetition and entity coverage in the generated texts.

Acknowledgments

This research was supported by the Office of Naval Research under the MURI grant N00014-18-1-2670, NSF (IIS 1616112, III 1703166), Allen Distinguished Investigator Award, Samsung GRO and gifts from Allen Institute for AI, Google, Amazon, and Bloomberg. We gratefully acknowledge the support of the European Research Council (Lapata; award number 681760). We also thank the anonymous reviewers and the UW-NLP group for their helpful comments.

References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the Literature Graph in Semantic Scholar. In *NAACL*.

Regina Barzilay and Mirella Lapata. 2005. Collective Content Selection for Concept-to-Text Generation. In *EMNLP*, pages 331–338. Association for Computational Linguistics.

Daniel Edward Robert Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *ACL*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*.

Michael J. Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Workshop on Statistical Machine Translation*.

Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks. *arXiv preprint arXiv:1211.3711*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Joohyun Kim and Raymond J Mooney. 2010. Generative Alignment and Semantic Parsing for Learning from Ambiguous Supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Svetlana Kiritchenko and Saif Mohammad. 2016. Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best-Worst Scaling. In *NAACL-HLT*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke S. Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *ACL*.

Ioannis Konstas and Mirella Lapata. 2013. Inducing Document Plans for Concept-to-Text Generation. In *EMNLP*, pages 1503–1514.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *EMNLP*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning Semantic Correspondences with Less Supervision. In *ACL/AFNLP*, pages 91–99.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press.

Jordan J Louviere and George G Woodworth. 1991. Best-Worst Scaling: A Model for the Largest Difference Judgments. *University of Alberta: Working Paper*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *EMNLP*, pages 3219–3232.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep Graph Convolutional Encoders for Structured Data to Text Generation. *INLG*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *NAACL-HLT*, pages 720–730.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*.

Lutz Prechelt. 1998. Early Stopping - but when? In *Neural Networks: Tricks of the Trade*, pages 55–69, London, UK, UK. Springer-Verlag.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-Text Generation with Content Selection and Planning. In *AAAI*.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12 1:145–151.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. Accessed at https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the Point: Summarization with Pointer-Generator Networks. *arXiv preprint arXiv:1704.04368*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A Graph-to-Sequence Model for AMR-to-Text Generation. In *ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. *arXiv preprint arXiv:1601.04811*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NeurIPS*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

Qingyun Wang, Zhihao Zhou, Lifu Huang, Spencer Whitehead, Boliang Zhang, Heng Ji, and Kevin Knight. 2018. Paper Abstract Writing through Editing Mechanism. In *Proceedings of NAACL-HLT*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-document Generation. In *EMNLP*.