Journal of Artificial Intelligence Research 70 (2021) 545-566 Submitted 04/2020; published 01/2021

An External Knowledge Enhanced Graph-Based Neural Network for Sentence Ordering

Yongjing Yin YINYONGJING@STU.XMU.EDU.CN
Xiamen University
Xiamen, China
Peng Cheng Laboratory
Shenzhen, China

Shaopeng Lai SPLAI@STU.XMU.EDU.CN
Xiamen University
Xiamen, China

Linfeng Song FREESUNSHINE0316@GMAIL.COM
Tencent AI Lab
Washington, USA

Chulun Zhou CLZHOU@STU.XMU.EDU.CN
Xiamen University
Xiamen, China

Xianpei Han XIANPEI@ISCAS.AC.CN
Institute of Software, Chinese Academy of Sciences
Beijing, China

Junfeng Yao YAO0010@XMU.EDU.CN
Xiamen University
Xiamen, China

Jinsong Su JSSU@XMU.EDU.CN
Xiamen University
Xiamen, China
Peng Cheng Laboratory
Shenzhen, China

Abstract

As an important text coherence modeling task, sentence ordering aims to coherently organize a given set of unordered sentences. To achieve this goal, the most important step is to effectively capture and exploit global dependencies among these sentences. In this paper, we propose a novel and flexible external knowledge enhanced graph-based neural network for sentence ordering. Specifically, we first represent the input sentences as a graph, where various kinds of relations (i.e., entity-entity, sentence-sentence and entity-sentence) are exploited to make the graph representation more expressive and less noisy. Then, we introduce graph recurrent network to learn semantic representations of the sentences. To demonstrate the effectiveness of our model, we conduct experiments on several benchmark datasets. The experimental results and in-depth analysis show our model significantly outperforms the existing state-of-the-art models.

1. Introduction

Modeling discourse coherence is crucial due to its significance to both natural language generation and understanding, which involves modeling logical consistency and topic transitions. Among various tasks of modeling text coherence, sentence ordering (Barzilay & Lapata, 2008) aims to restore the original coherent paragraph from a set of unordered sentences, which is a common approach to building and evaluating text coherence models. For many natural language processing (NLP) tasks where the relationships between sentences play an essential role, such as text generation (Konstas & Lapata, 2012; Holtzman et al., 2018), retrieval-based question answering (Yu et al., 2018), conversational analysis (Zeng et al., 2018), essay scoring (Burstein et al., 2010), and multi-document summarization (Barzilay et al., 2002; Galanis et al., 2012; Nallapati et al., 2012), sentence order is of great significance because the final performance will be negatively affected by erroneous sentence orders. Concretely, in multi-document summarization, since the relative ordering of sentences from different documents can be unclear, finding acceptable orderings can enhance user comprehension (Barzilay et al., 2002). In brief, sentence ordering is a fundamental NLP task and can be easily extended and applied into various downstream tasks. A successful sentence ordering model is not only able to understand the discourse structure, but also help to generate well-organized long text to achieve both local and global consistency. Therefore, sentence ordering has attracted increasing attention in recent years.

To achieve this goal, in the early studies, researchers mainly resorted into rule-based or statistical approaches. However, these approaches often involve careful designs of various sophisticated linguistic features, which often require high labor costs. Recently, with the rapid development and wide applications of deep learning, dominant sentence ordering models have evolved into neural network based ones. In this aspect, the typical models mainly consist of window network (Li & Hovy, 2014; Li & Jurafsky, 2017), hierarchical RNN-based model (Logeswaran et al., 2018), and deep attentive sentence ordering network (*ATTOrderNet*) (Cui et al., 2018). In particular, taking advantages of fully-connected graph representation based multi-head self-attention in learning sentence representations, *ATTOrderNet* achieves the state-of-the-art performance in the field of sentence ordering.

However, *ATTOrderNet* still has two serious defects. On the one hand, although its fully-connected graph representations allow the whole model to encode relationship between sentences, lots of noises caused by connecting incoherent sentences are also introduced, resulting in a negative effect on the encoder. On the other hand, although the self-attention mechanism have shown effective in modeling text coherence, it only exploits sentence-level information using the same set of parameters while ignoring extra information such as entities. Therefore, exploring a more effective encoder for neural sentence ordering is still challenging.

In this paper, we propose a novel external knowledge enhanced graph-based neural network for sentence ordering, which represents the input sentences as a graph and then adopts recent graph recurrent network (GRN) (Zhang et al., 2018b) to learn sentence representations. Specifically, we explore two graphs to represent input sentences and their entities: one is sentence-entity graph proposed by Guinaudeau and Strube (2013), the other is the extension of the sentence-entity graph, where external lexical knowledge is exploited to enrich the graph structure. In our graphs, each node denotes either a sentence or an en-

S1: Maria looked at the ***menu*** and ordering some ***food***.

S3: Maria added the ***item*** to the ***menu***.

S2: She decided that she wanted an ***order*** of a ***salad***.

S4: She paid for the ***item*** and placed the ***order*** for pickup.

Figure 1: An example of sentence ordering, where the correct order is: S1, S2, S3, S4. S3 is coherent with S4, as they share the same entity "*item*". S1 and S2 are coherent, since the entity "*menu*" is closely related to the entity "*order*".

tity, and each edge either links two semantically related sentence nodes or entity nodes, or connects one entity node with its corresponding sentence node. In this way, our graph representations can effectively encode the following information: the semantic relevance between coherent sentences, the co-occurrence between sentences and entities, and the lexical relation between entities.

To better illustrate the intuition behind our graph representations, we take the sentences shown in Figure 1 as example. Here we can obtain the following observations: First, if two sentences contain the same entity, they are likely to be semantically close to each other: the entity "*item*" occurs in both the sentences S3 and S4, which are obviously more coherent than S1 and S4. Second, the entity "*menu*" is closely related to the entity "*order*", which provides an important clue that S3 may appear behind S2. Overall, compared with the fully-connected graph representation of *ATTOrderNet* (Cui et al., 2018), our graph has two obvious advantages: First, it is able to greatly reduce the noisy edges between irrelevant sentence nodes. Second, the rich and useful sentence-sentence, entity-entity, and entity-sentence information can be fully leveraged to refine paragraph encoding.

On the basis of sentence-entity graph representations, we then introduce GRN (Zhang et al., 2018b) to conduct paragraph encoding by recurrently performing semantic transitions among connected nodes. During this process, we use a paragraph-level node to assemble semantic information of all nodes. By doing so, the resulting representation of this node will be beneficial to the long-distance information propagation among nodes. Besides, considering the fact that sentence nodes and entity nodes play different roles, we use different parameters to distinguish and quantify their effects. After learning the paragraph representation, we also use a pointer network decoder to generate the ordered sentence sequence.

Overall, in this work, our main contributions can be summarized as follows:

- We represent the input unordered sentences as a sentence-entity graph, which extends the graph presentation proposed by Guinaudeau and Strube (2013) by exploiting lexical relations between entities.

- We introduce GRN-based encoder based on the above-mentioned graph representations to learn sentence and paragraph embeddings. To the best of knowledge, our work is the first attempt to explore such a graph-based encoder for sentence ordering.

- Experimental results and in-depth analysis demonstrate the effectiveness of our proposed encoder, particularly, verifying the effectiveness of entities in graph representations for sentence ordering.

This work has been presented in our previous conference paper (Yin et al., 2019). In this paper, we significantly extend our method from previous work in the following aspects:

- We introduce external lexical knowledge to enhance the graph-based model. Specifically, nodes of semantically related entities are connected, which making the extended graph more expressive. Moreover, the entity relatedness induced from lexical knowledge as priori weight coefficients leads to better paragraph encoding for sentence ordering.

- We carry out more experiments and analysis to further investigate the effectiveness of our model in multi-document extractive summarization.

The remainder of this paper is organized as follows. Section 2 summarizes the related work and highlights the differences of our model from previous studies; Section 3 gives a brief description to our baseline model, *ATTOrderNet*; Section 4 elaborates on details of our proposed model; Experimental results are reported and analyzed in Section 5; Finally, we conclude with some remarks on prospective future directions in Section 6.

2. Related Work

In this work, we propose a graph-based neural network for sentence ordering. Our related work mainly includes two aspects:

Sentence Ordering Previous work on sentence ordering mainly focused on the utilization of linguistic features via statistical models (Lapata, 2003; Barzilay & Lee, 2004; Barzilay & Lapata, 2005, 2008; Elsner & Charniak, 2011; Guinaudeau & Strube, 2013). Especially, the entity based models (Barzilay & Lapata, 2005, 2008; Guinaudeau & Strube, 2013) have shown the effectiveness of exploiting entities for this task. Recently, with the rapid development of deep learning, the studies have evolved into neural network based models. Li and Jurafsky (2017) describe both discriminative and generative neural models that are able to measure coherence in existing sentences, marking an initial step in generating coherent texts given discourse contexts. Logeswaran et al. (2018) propose a hierarchical RNN-based model based on set-to-sequence framework. Cui et al. (2018) introduce the self-attention mechanism to modeling input sentences. For neural coherence models used to assess text coherence, Nguyen and Joty (2017) propose a neural version of the entity grid model. Mohiuddin et al. (2018) further use neural entity grid to model the coherence of conversations. More recently, Moon et al. (2019) incorporate local and global coherence model into a unified framework.

The above neural sentence ordering models do not fully exploit inter-sentence coherence relations and entity information which have proven highly useful in formal theories of discourse. In addition, they only implicitly learn some lexical knowledge form corpora and do not utilize the external knowledge base (e.g. WordNet) to provide more explicit lexical knowledge. In this work, we effectively combine the advantages of modeling entity

information and structure encoding of graph neural networks, achieving state-of-the-art performance.

Graph Neural Networks in NLP Recently, graph neural networks have proven to be effective in many NLP tasks, such as modeling semantic graphs (Beck et al., 2018; Song et al., 2018a, 2019), dependency trees (Marcheggiani & Titov, 2017a; Bastings et al., 2017; Vashishth et al., 2018; Zhang et al., 2018c; Song et al., 2018b), knowledge graphs (Wang et al., 2018), and even multi-modal input (Yin et al., 2020b). Specially, our work is inspired by previous studies that incorporate prior knowledge into graph neural networks. For example, Marcheggiani and Titov (2017b) introduce syntactic knowledge by parsing sentences into dependency graphs and then model it by GCN. Li et al. (2019) extract key words of sentences as topic knowledge, and organize the article into a topic interaction graph. Vashishth et al. (2019) propose SynGCN which incorporates various semantic relations (e.g. hyponymy, hypernymy and synonymy) when constructing the graph. Sahu et al. (2019) introduce coreference relations as extra knowledge and connects co-referred entities in a document-level graph.

In addition, our work is in line with GRN (Zhang et al., 2018b), which achieves excellent performances on many text classification and sequence labeling tasks. In our work, we extend GRN from encoding sentences to encoding paragraphs. Moreover, we further exploit external knowledge to enrich graph representations, leading to better sentence ordering.

3. Baseline: *ATTOrderNet*

In this section, we give a brief introduction to *ATTOrderNet*, proposed by Cui et al. (2018). Due to its satisfying performance, *ATTOrderNet* has been widely used and thus selected as the baseline of our work. Taking a set of M input sentences $\boldsymbol{s} = \{s_{o_1}, \ldots, s_{o_M}\}$ as input, *ATTOrderNet* aims to recover the correct order $\boldsymbol{o}^* = [o_1^*, \ldots, o_M^*]$. Figure 2 shows the architecture of ATTOrderNext. Overall, it consists of three components: (1) a Bi-LSTM sentence encoder, (2) a paragraph encoder based on multi-head self-attention (Vaswani et al., 2017), and (3) a pointer network based decoder (Vinyals et al., 2015).

3.1 Bi-LSTM Based Sentence Encoder

The Bi-LSTM sentence encoder reads the word embedding sequence $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ of the input sentence s_{o_i} in two directions, producing its bidirectional hidden state sequences. Specifically, at the j-th timestep, the current states ($\overrightarrow{\boldsymbol{h}}_j$ and $\overleftarrow{\boldsymbol{h}}_j$) are produced from the previous hidden states ($\overrightarrow{\boldsymbol{h}}_{j-1}$ and $\overleftarrow{\boldsymbol{h}}_{j+1}$) and the current word embedding $\boldsymbol{x}_j$ in the following way:

$$\overrightarrow{\boldsymbol{h}}_j = \text{LSTM}(\overrightarrow{\boldsymbol{h}}_{j-1}, \boldsymbol{x}_j), \tag{1}$$

$$\overleftarrow{\boldsymbol{h}}_j = \text{LSTM}(\overleftarrow{\boldsymbol{h}}_{j+1}, \boldsymbol{x}_j). \tag{2}$$

Finally, we concatenate the last states of the Bi-LSTM in both directions to denote the sentence representation of sentence s_{o_i} i.e. $\boldsymbol{\kappa}_{o_i}^0 = [\overrightarrow{\boldsymbol{h}}_n; \overleftarrow{\boldsymbol{h}}_1]$.

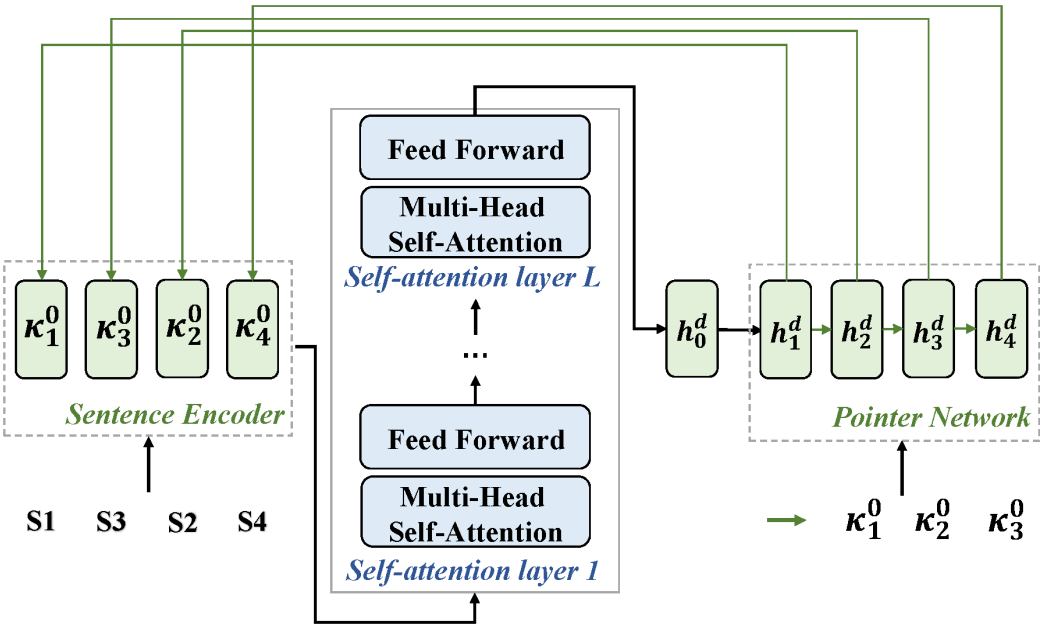

Figure 2: The architecture of *ATTOrderNet* (Cui et al., 2018).

3.2 Multi-head Self-Attention Based Paragraph Encoder

On the top of the above-mentioned Bi-LSTM sentence representations, the paragraph encoder is equipped with several self-attention layers followed by an average pooling layer to refine these sentence representations.

Specifically, the paragraph representation is first initialized by concatenating all sentence representations, i.e, $\boldsymbol{K}^0 = [\boldsymbol{\kappa}^0_{o_1}; \ldots; \boldsymbol{\kappa}^0_{o_M}]$. Then, this initial graph representation is updated by L self-attention layers, where the update at the l-th layer is conducted as follows:

$$\boldsymbol{K}^l = \text{SelfAttention}_l(\boldsymbol{K}^{l-1}), \tag{3}$$

where SelfAttention_l consists of a multi-head self-attention and a feed-forward network. Finally, an average pooling operation is performed on the output K^L of the last self-attention layer to produce the final paragraph representation $\boldsymbol{g}$:

$$\boldsymbol{g} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\kappa}^L_i, \tag{4}$$

where $\boldsymbol{\kappa}^L_i$ denotes the final vector representation of sentence s_{o_i}.

3.3 Pointer Network Based Decoder

With the learned final paragraph representation $\boldsymbol{g}$, a pointer network based decoder is introduced to generate the ordered sentence sequence. Formally, the conditional probability

of the input paragraph $\boldsymbol{s}$ with a predicted order $\boldsymbol{o'}$ is defined as follows:

$$P(\boldsymbol{o'}|\boldsymbol{s}) = \prod_{i=1}^{M} P(o'_i|\boldsymbol{o'}_{<i}, \boldsymbol{s}), \tag{5}$$

$$P(o'_i|\boldsymbol{o'}_{<i}, \boldsymbol{s}) = \text{Softmax}(\boldsymbol{v}^{\intercal} \tanh(\boldsymbol{W}\boldsymbol{h}_i^d + \boldsymbol{U}\boldsymbol{K}^0)), \tag{6}$$

where $\boldsymbol{h}_i^d$ is the decoder state, and $\boldsymbol{v}$, $\boldsymbol{W}$ and $\boldsymbol{U}$ are learned model matrices.

During the model training, the input sequence of the decoder is the vector representation $\left[\boldsymbol{\kappa}^0_{o^*_1}, \ldots, \boldsymbol{\kappa}^0_{o^*_M}\right]$ with the correct sentence order $\boldsymbol{o}^*$. At each timestep of the model testing, the input of the decoder corresponds to the sentence representation in the predicted order, and the decoder state is updated recurrently in the follow way:

$$\boldsymbol{h}_i^d = \text{LSTM}(\boldsymbol{h}_{i-1}^d, \boldsymbol{\kappa}^0_{o'_{i-1}}), \tag{7}$$

where $\boldsymbol{k}^0_{o'_{i-1}}$ denotes the vector representation of the previously ordered sentence. Particularly, $\boldsymbol{h}_0^d$ is initialized as the final paragraph representation $\boldsymbol{g}$, and the first-step input and initial cell memory are zero vectors.

4. Our Model

In this section, we introduce our graph-based neural sentence ordering model in detail. Overall, our proposed model mainly includes a sentence encoder, a GRN based paragraph encoder and a pointer network based decoder. In order to ensure fair comparison, we directly adapt the sentence encoder and decoder from ATTOrderNet (Cui et al., 2018). Therefore, we omit the descriptions of the encoder and decoder, and mainly focus on the description of our paragraph encoder, involving graph representations and GRN-based paragraph encoding.

4.1 Graph Representations

Initially, we need to represent an input paragraph as a graph. Different from the fully-connected graph representations explored by Cui et al. (2018), we leverage entity information to construct suitable graph representations for the subsequent graph-based paragraph encoding. In our graphs, the entities not only provide important clues for modeling text coherence, but also can be used to alleviate the noise caused by connecting incoherent sentences. Moreover, we introduce the word-level relatedness information from external knowledge base to enrich our graph.

To construct our graphs, we first regard all nouns in each input paragraph as entities, and remove the entities that only appear once in the paragraph so as to alleviate the negative effect of redundant entities in long paragraphs. Based on the identified entities, we then explore two types of graph representations for neural sentence ordering, which will be described in detail.

SE-Graph Following Guinaudeau and Strube (2013), we first use sentence-entity graphs to represent input paragraphs. As shown in Figure 3 (b), the SE-Graph can be formalized as $\boldsymbol{G_{se}} = (\boldsymbol{V_s}, \boldsymbol{V_e}, \boldsymbol{\mathcal{E}_{se}})$, where $\boldsymbol{V_s}$, $\boldsymbol{V_e}$ and $\boldsymbol{\mathcal{E}_{se}}$ indicate the sentence-level nodes (such as

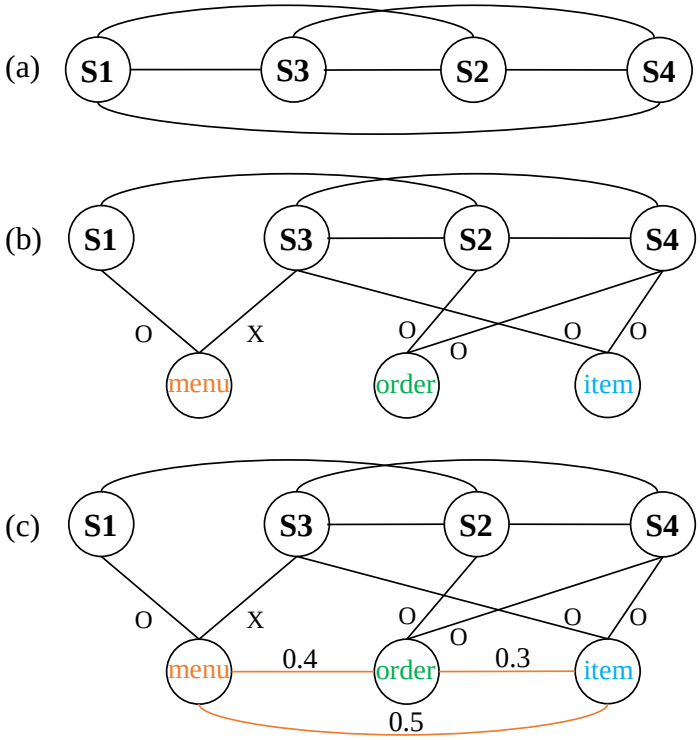

Figure 3: Comparison among (a) a fully-connected graph used by Cui et al. (2018), (b) our sentence-entity graph, and (c) our sentence-entity graph with external lexical knowledge for the example in Figure 1. An edge label in (b) and (c) corresponds to the syntactic role of an entity in a sentence. Compared to the graph used in (Yin et al., 2019), nodes of semantically related entities are connected (the edges in orange), which making the extended graph more expressive. Moreover, the entity relatedness induced from lexical knowledge as priori weight coefficients leads to better paragraph encoding.

v_{s_i}), entity-level nodes (such as v_{e_j}), and edges, respectively. There are two types of edges in $\boldsymbol{\mathcal{E}_{se}}$. The first type of edge is sentence-entity edge connecting a sentence and an entity in it. It has a label indicating the syntactic role of the entity in the sentence (Guinaudeau & Strube, 2013), which can be either a subject(S), an object(O) or other(X). Particularly, since an entity has different roles in the same sentence, we choose the highest-rank role as its final label according to the rule $S \succ O \succ X$. The second type of edge is sentence-sentence edge linking two sentences that have common entities. Consequently, sentence nodes can be linked to sentence or entity nodes, and entity nodes are only connected to sentence ones.

SEK-Graph This kind of graph is an extension of the first one, which exploits the word relatedness induced from external knowledge base to enrich the graph representation. Likewise, it can also be formalized as $\boldsymbol{G_{sek}} = (\boldsymbol{V_s}, \boldsymbol{V_e}, \boldsymbol{\mathcal{E}_{sek}})$, where the node sets are same as those in $\boldsymbol{G_{se}}$, and the edge set $\boldsymbol{\mathcal{E}_{sek}}$ has an additional type of entity-entity edges connecting related entity nodes. In particular, each edge has a weight coefficient that indicates the semantic relevance between two entities. Intuitively, in a paragraph, an entity is often related to its synonyms or hypernyms, which can provide important clues to text coherence.

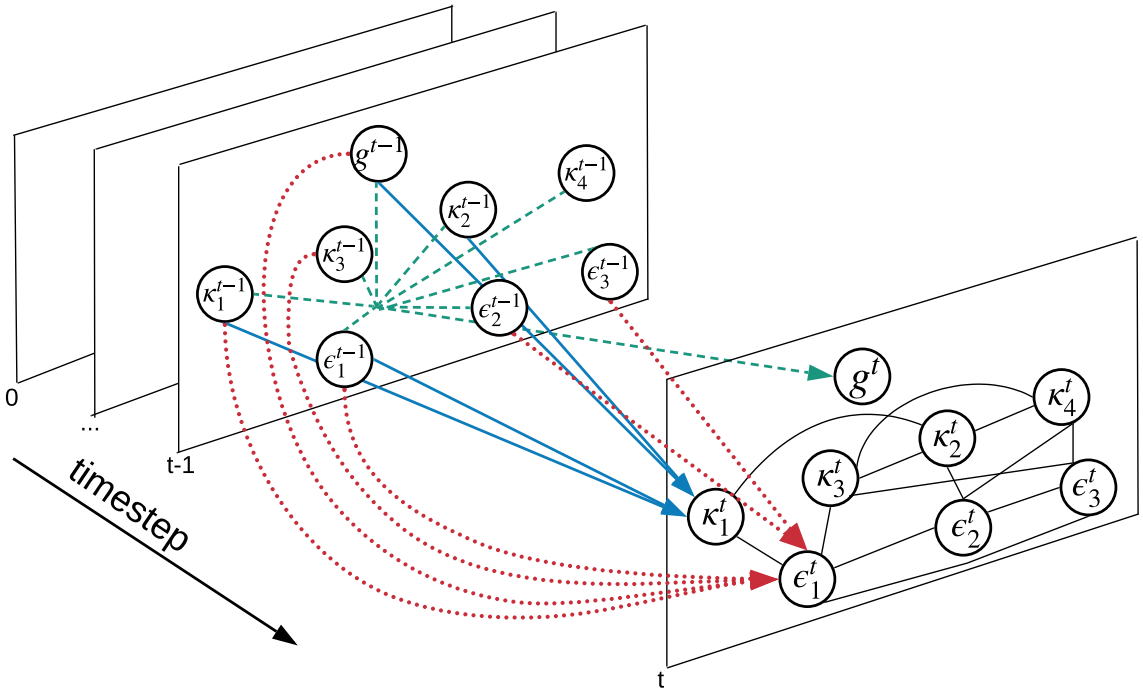

Figure 4: GRN encoding our proposed SEK-Graph. Please note that GRN encoding for SE-Graph is similar to that for SEK-Graph, with the difference that each entity node will not receive information from other entity nodes. The original graph structure is only drawn on timestep t for being concise.

Therefore, we utilize the commonly-used toolkit[1] to calculate the word-level relatedness between two entities based on WordNet, and connect two entity nodes of which the word-level relatedness is larger than a threshold.

To better understand our proposed graph representations and previous one (Cui et al., 2018), we compare them in Figure 3. In the fully-connected graph, the edges are dense, which may introduce noise especially when the input paragraph contains abundant sentences. For example, S1 and S4 do not share any common entities and thus we believe that they do not form a coherent sentence sequence. However, there still exists one edge connecting them in this fully-connected graph. In contrast, both SE-Graph and SEK-Graph do not contain this edge, avoiding the resulting noise. Another problem with the fully-connected graph is that every node is directly linked to others, thus no information can be obtained based on this kind of graph structure. In contrast, the structure of our sentence-entity graphs can provide more discriminating information. Particularly, SEK-Graph encodes the richer information between entities induced from external knowledge base.

1. http://code.google.com/archive/p/xssm/downloads

4.2 Encoding with GRN

Based on the above graph representations, we construct the paragraph encoder using GRN, which has shown effective in many tasks such as text representation (Zhang et al., 2018b) and text generation (Song et al., 2019; Beck et al., 2018). Essentially, it is a kind of graph neural network based on a message passing framework, and node states are updated iteratively using recurrent gating mechanisms such as LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Cho et al., 2014). At the t-th step of message passing, the update process of each node involves two sub-steps: the message is first collected from its adjacent nodes, and then is used to update the node state using recurrent gating mechanisms. In our work, we choose GRU to model state transitions since it has fewer number of parameters and better efficiency.

Figure 4 shows the message passing procedures of our paragraph encoders based on our proposed different graph representations. Although the proposed graph representations are different, their resulting encoders are similar, where the only difference lies in the update of entity nodes. In addition to all node states in the graph, we introduce a paragraph-level state $\boldsymbol{g}$, which can provide global information for the updates of both sentence and entity nodes. Since sentence nodes and entity nodes have different roles and contain different amount of semantic information, we apply separate encoding parameters and gating operations to model their state transition processes.

At the t-th step, we update the sentence state $\boldsymbol{\kappa}_i^{(t-1)}$ of the node v_{s_i} using the weighted sum with messages from its neighboring sentence states and entity states. Concretely, we first collect two kinds of information from its neighboring sentence nodes and entity nodes as follows:

$$\boldsymbol{m}_i^{(t)} = \sum_{v_{s_i'} \in N(v_{s_i})} \boldsymbol{w}_{i,i'}^{(t)} \boldsymbol{\kappa}_{i'}^{(t-1)}, \tag{8}$$

$$\tilde{\boldsymbol{m}}_i^{(t)} = \sum_{v_{e_j} \in \tilde{N}(v_{s_i})} \tilde{\boldsymbol{w}}_{i,j}^{(t)} \boldsymbol{\epsilon}_j^{(t-1)} \tag{9}$$

where $N(v_{s_i})$ and $\tilde{N}(v_{s_i})$ are the neighboring sentence and entity nodes sets of v_{s_i}, respectively. The weights $\boldsymbol{w}_{i,j}^{(t)}$ and $\tilde{\boldsymbol{w}}_{i,j}^{(t)}$ are computed according to their edge label $l_{i,j}$ (if any) and two associated node states, both of which are fed into a single-layer neural network with a sigmoid activation function. Then, we update the sentence state $\boldsymbol{\kappa}_i^{(t-1)}$ using $\boldsymbol{m}_i^{(t)}$, $\tilde{\boldsymbol{m}}_i^{(t)}$ and $\boldsymbol{g}^{(t-1)}$ in the following way:

$$\boldsymbol{\xi}_i^{(t)} = [\boldsymbol{s}_i; \boldsymbol{m}_i^{(t)}; \tilde{\boldsymbol{m}}_i^{(t)}; \boldsymbol{g}^{(t-1)}] \tag{10}$$

$$\boldsymbol{r}_i^{(t)} = \sigma\left(\boldsymbol{W}^r \boldsymbol{\xi}_i^{(t)} + \boldsymbol{U}^r \boldsymbol{\kappa}_i^{(t-1)}\right) \tag{11}$$

$$\boldsymbol{z}_i^{(t)} = \sigma\left(\boldsymbol{W}^z \boldsymbol{\xi}_i^{(t)} + \boldsymbol{U}^z \boldsymbol{\kappa}_i^{(t-1)}\right) \tag{12}$$

$$\boldsymbol{u}_i^{(t)} = tanh\left(\boldsymbol{W}^u \boldsymbol{\xi}_i^{(t)} + \boldsymbol{U}^u \left(\boldsymbol{r}_i^{(t)} \odot \boldsymbol{\kappa}_i^{(t-1)}\right)\right) \tag{13}$$

$$\boldsymbol{\kappa}_i^{(t)} = \left(\boldsymbol{1} - \boldsymbol{z}_i^{(t)}\right) \odot \boldsymbol{u}_i^{(t)} + \boldsymbol{z}_i^{(t)} \odot \boldsymbol{\kappa}_i^{(t-1)} \tag{14}$$

Please note that the update processes of entity nodes are different in SE-Graph and SEK-Graph encoding. Specifically, for an entity node v_{e_j} in SE-Graph, its state $\boldsymbol{\epsilon}_j^{(t-1)}$ is updated based on its word embedding $\boldsymbol{e}_j$, the states of its connected sentence nodes (such as $\boldsymbol{\kappa}_i^{(t-1)}$) and $\boldsymbol{g}^{(t-1)}$. Similarly, we first gather the information from its neighboring nodes:

$$\hat{\boldsymbol{m}}_j^{(t)} = \sum_{v_{s_i} \in N(v_{e_j})} \hat{\boldsymbol{w}}_{j,i}^{(t)} \boldsymbol{\kappa}_i^{(t-1)}, \tag{15}$$

$$\hat{\boldsymbol{\xi}}_j^{(t)} = [\boldsymbol{e}_j; \hat{\boldsymbol{m}}_j^{(t)}; \boldsymbol{g}^{(t-1)}], \tag{16}$$

where $N(v_{e_j})$ denotes the sentence-level neighbors of an entity node v_{e_j}.

In SEK-Graph, since an entity node may link to other entity nodes, we collect additional message from its connected entity nodes for state transition. Particularly, inspired by previous studies incorporating prior knowledge, we leverage the word-level semantic relatedness between entities to guide the updates of entity nodes:

$$\tilde{\boldsymbol{m}}_j^{(t)} = \sum_{v_{e'_j} \in \tilde{N}(v_{e_j})} \bar{\boldsymbol{w}}_{j,j'}^{(t)} \boldsymbol{\epsilon}_{j'}^{(t-1)}, \tag{17}$$

$$\bar{\boldsymbol{w}}_{j,j'}^{(t)} = \sigma\left(\boldsymbol{W}_1 \boldsymbol{\epsilon}_j^{(t-1)} + \boldsymbol{W}_2 \boldsymbol{\epsilon}_{j'}^{(t-1)} + \lambda \odot Rele(v_{e_j}, v_{e_{j'}})\right), \tag{18}$$

$$\hat{\boldsymbol{\xi}}_j^{(t)} = [\boldsymbol{e}_j; \hat{\boldsymbol{m}}_j^{(t)}; \tilde{\boldsymbol{m}}_j^{(t)}; \boldsymbol{g}^{(t-1)}], \tag{19}$$

where $\tilde{N}(v_{e_j})$ denotes the set of adjacent entity nodes of v_{e_j}, and $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and λ are trainable parameters. Here, $Rele(v_{e_j}, v_{e_{j'}})$ is the semantic relatedness between two entities derived from WordNet, which is used as a priori weight coefficient.

After obtaining all messages, the new entity state $\boldsymbol{\epsilon}_j^{(t)}$ is generated using GRU:

$$\hat{\boldsymbol{r}}_j^{(t)} = \sigma\left(\hat{\boldsymbol{W}}^r \hat{\boldsymbol{\xi}}_j^{(t)} + \hat{\boldsymbol{U}}^r \boldsymbol{\epsilon}_j^{(t-1)}\right) \tag{20}$$

$$\hat{\boldsymbol{z}}_j^{(t)} = \sigma\left(\hat{\boldsymbol{W}}^z \hat{\boldsymbol{\xi}}_j^{(t)} + \hat{\boldsymbol{U}}^z \boldsymbol{\epsilon}_j^{(t-1)}\right) \tag{21}$$

$$\hat{\boldsymbol{u}}_j^{(t)} = tanh\left(\hat{\boldsymbol{W}}^u \hat{\boldsymbol{\xi}}_j^{(t)} + \hat{\boldsymbol{U}}^u \left(\hat{\boldsymbol{r}}_j^{(t)} \odot \boldsymbol{\epsilon}_j^{(t-1)}\right)\right) \tag{22}$$

$$\boldsymbol{\epsilon}_j^{(t)} = \left(\boldsymbol{1} - \hat{\boldsymbol{z}}_j^{(t)}\right) \odot \hat{\boldsymbol{u}}_j^{(t)} + \hat{\boldsymbol{z}}_j^{(t)} \odot \boldsymbol{\epsilon}_j^{(t-1)} \tag{23}$$

Afterwards, we update the state of paragraph-level node $\boldsymbol{g}^{(t-1)}$ with messages from all sentence nodes and entity nodes:

$$\hat{\boldsymbol{\kappa}}^{(t-1)} = \frac{1}{|V_s|}\sum_{m=1}^{|V_s|}\boldsymbol{\kappa}_m^{(t-1)} \tag{24}$$

$$\hat{\boldsymbol{\epsilon}}^{(t-1)} = \frac{1}{|V_e|}\sum_{m=1}^{|V_e|}\boldsymbol{\epsilon}_m^{(t-1)} \tag{25}$$

$$\boldsymbol{r}_g^{(t)} = \sigma(\boldsymbol{W}^{sr}\hat{\boldsymbol{\kappa}}^{(t-1)} + \boldsymbol{W}^{er}\hat{\boldsymbol{\epsilon}}^{(t-1)} + \boldsymbol{U}^{gr}\boldsymbol{g}^{(t-1)}) \tag{26}$$

$$\boldsymbol{z}_g^{(t)} = \sigma(\boldsymbol{W}^{sz}\hat{\boldsymbol{\kappa}}^{(t-1)} + \boldsymbol{W}^{ez}\hat{\boldsymbol{\epsilon}}^{(t-1)} + \boldsymbol{U}^{gz}\boldsymbol{g}^{(t-1)}) \tag{27}$$

$$\boldsymbol{u}_g^{(t)} = tanh(\boldsymbol{W}^{su}\hat{\boldsymbol{\kappa}}^{(t-1)} + \boldsymbol{W}^{eu}\hat{\boldsymbol{\epsilon}}^{(t-1)} + \boldsymbol{U}^{gu}(\boldsymbol{r}_g^{(t)} \odot \boldsymbol{g}^{(t-1)})) \tag{28}$$

$$\boldsymbol{g}^{(t)} = (\mathbf{1} - \boldsymbol{z}_g^{(t)}) \odot \boldsymbol{u}_g^{(t)} + \boldsymbol{z}_g^{(t)} \odot \boldsymbol{g}^{(t-1)}, \tag{29}$$

where $\boldsymbol{W}^*$ $(* \in \{r, z, u, sr, er, sz, ez, su, eu\})$, $\boldsymbol{U}^*$ $(* \in \{r, z, u, gr, gz, gu\})$, $\hat{\boldsymbol{W}}^*$ and $\hat{\boldsymbol{U}}^*$ $(* \in \{r, z, u\})$ denote trainable parameters, and $|V_s|$ and $|V_e|$ are the number of sentences and entities respectively. In this way, each node state not only absorbs contextual information through the iterative encoding process but captures co-occurrence relationships between sentences and entities.

The above update process will iterate T times. Finally, we obtain the final paragraph state $\boldsymbol{g}^{(T)}$, which is used to initialize the initial decoder state $\boldsymbol{h}_0^d$ (see Section 3) to predict an ordered sentence sequence.

5. Experiments

To investigate the effectiveness of our model, we compare it with several state-of-the-arts using four commonly-used sentence ordering datesets. Besides, we further demonstrate the validity of our model in the task of multi-document summarization.

5.1 Setup

Datasets. We conduct experiments on the following benchmark datasets:

- **NIPS Abstract**. This dataset contains roughly 3K abstracts of NIPS papers from 2005 to 2015.
- **ANN Abstract**. It includes about 12K abstracts extracted from the papers in ACL Anthology Network (AAN) (Radev et al., 2016).
- **arXiv Abstract**. This dataset comes from arXiv and it consists of about 1.1M abstracts.
- **SIND**. It has 50K stories for the visual storytelling task[2], which belongs to a different domain. Here we use each story as a paragraph.

2. http://visionandlanguage.net/VIST/

We first use *NLTK*[3] to tokenize the sentences, and then adopt *Stanford Parser*[4] to extract nouns with syntactic roles for the edge labels (S, O or X). For each paragraph in the first three datasets in which there are lots of nouns, we only treat the nouns appearing more than once as entities. In this way, each paragraph from NIPS Abstract, ANN Abstract, arXiv Abstract and SIND has 5.8, 4.5, 7.4 and 9.1 entities on average, respectively.

Settings. To ensure fair comparisons, we use the same setting as (Cui et al., 2018). Specifically, we first apply *Glove* to pre-train 100-dimensional word embeddings, which will be fine-tuned during the model training. Besides, to construct our GRN encoder, we set the state sizes for sentence and entity nodes to 512 and 150, respectively, and the size of edge embedding to 50. As for the model training, we adopt *Adadelta* (Zeiler, 2012) with $\epsilon = 10^{-6}$, $\rho = 0.95$ and initial learning rate 1.0 as the optimizer. Particularly, we employ *L2* weight decay with the coefficient 10^{-5} and *dropout* with the probability 0.5 for regularization term. Batch size is set to 16 for training and beam search with size 64 is implemented for decoding. The optimal thresholds chosen on validation sets for NIPS Abstract, ANN Abstract, arXiv Abstract and SIND are 0.6, 0.6, 0.6 and 0.7 respectively.

Contrast Models. We still directly refer to two variants of our models that encode our two proposed graph representations as ***SE-Graph*** and ***SEK-Graph*** respectively, and then compare them with the following contrast models: (1) ***LSTM+PtrNet*** (Gong et al., 2016), (2) ***Varient-LSTM+PtrNet*** (Logeswaran et al., 2018), and (3) ***ATTOrderNet*** (Cui et al., 2018). Note that major differences among these models lie in their ways of encoding paragraphs.

In addition to the above-mentioned contrast models, we also report the performance of two variants of our model, so as to investigate effects of various factors on our model: (1) ***F-Graph***. Similar to ATTOrderNet, it uses a fully-connected graph to represent the input unordered paragraph, but adopts GRN rather than self-attention layers to encode the graphs. (2) ***S-Graph***. It is a simplified version of our model by removing all entity nodes and their related edges from the original sentence-entity graphs. (3) ***FE-Graph***. It encodes the fully-connected graph of sentences as well as all entities in the input paragraph.

Evaluation Metrics. As implemented in previous work, we employ the following three metrics to directly measure the quality of reordered sentence sequences:

- **Kendall tau ($\boldsymbol{\tau}$)**: Formally, it is calculated as $\boldsymbol{\tau}$ = 1- 2×(*number of inversions*)/$\binom{M}{2}$, where M denotes the sequence length and *number of inversions* is the number of pairs in the predicted sequence with incorrect relative order. Note that it ranges from -1 (the worst) to 1 (the best).

- **Accuracy (Acc)**: It measures the percentage of sentences with correctly predicted absolute positions. Different from τ, it penalizes results that correctly preserve most relative orders but with a slight shift.

- **Perfect Match Ratio (PMR)**: It considers each paragraph as a single unit and calculates the ratio of exactly matching orders, so no partial credit is given for any incorrect permutations.

3. https://www.nltk.org/
4. https://nlp.stanford.edu/software/lex-parser.shtml

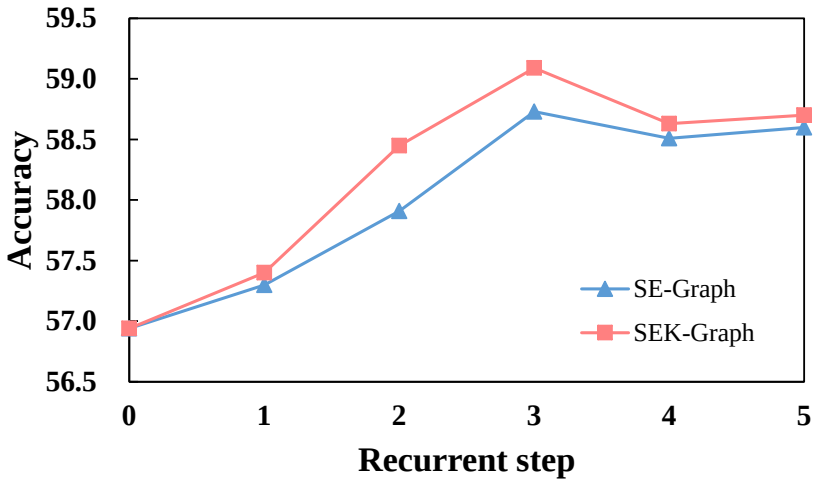

Figure 5: Results on the arXiv Abstract validation set regarding the recurrent step T.

5.2 Effect of Recurrent Step T

Obviously, the recurrent step T is a crucial hyperparameter in our models. Here we carry out experiments on the largest validation set (arXiv Abstract) to study its effect.

From the experimental results shown in Figure 5, we can find large improvements in all models when increasing T from 0 to 3, demonstrating the effectiveness of our proposed models. However, the increase of T from 3 to 5 does not bring further improvements while leading to more running time. Hence, we directly use T=3 for all experiments thereafter.

5.3 Main Results

The overall experimental results are shown in Table 1. Our SEK-Graph significantly outperforms other contrast models and its variants on the datasets of different domains. Through in-depth analyses, we can reach the following conclusions:

First, even with fewer number of parameters and relatively fewer recurrent steps, the F-Graph model based on the same fully-connected graph representations is still able to slightly surpass ATTOrderNet on all datasets. Therefore, we believe that it is reasonable to apply GRN for encoding paragraphs.

Second, S-Graph exhibits better performance than F-Graph. This confirms the hypothesis that leveraging entity information can reduce the noise caused by connecting incoherent sentences.

Third, SE-Graph outperforms S-Graph on all datasets. This result is consistent with our assumption that exploiting entities as extra information and modeling the co-occurrence between sentences and entities can further contribute to our neural graph model. Besides, SE-Graph is superior to FE-Graph and the performance is more stable, which shows that reducing the noise from fully-connected sentence nodes and rudantant entity nodes is better for model learning.

Fourth, SEK-Graph obtains better performance than SE-Graph, verifying the benefit of modeling related entities for paragraph encoding.

Model	NIPS Abstract			ANN Abstract		
	Acc	τ	#pm	Acc	τ	#pm
LSTM+PtrNet †	50.87	67.00	2.1M	58.20	69.00	3.0M
Varient-LSTM+PtrNet †	51.55	72.00	26.5M	58.06	73.00	28.9M
ATTOrderNet †	56.09	72.27	8.7M	63.24	73.40	17.9M
F-Graph	56.24	72.41	4.1M	63.45	74.24	9.9M
S-Graph	56.67	73.00	4.1M	64.09	76.08	9.9M
FE-Graph	55.13	74.80	5.0M	62.75	77.18	11.5M
SE-Graph	57.27*	75.10*	5.0M	64.64*	78.26*	11.5M
SEK-Graph	**58.25***	**76.49***	5.2M	**65.06***	**78.60***	12.0M
Model	arXiv Abstract			SIND		
	PMR	τ	#pm	PMR	τ	#pm
LSTM+PtrNet †	40.44	72.00	12.7M	12.34	48.00	3.6M
Varient-LSTM+PtrNet †	–	–	–	–	–	–
ATTOrderNet †	42.19	73.00	23.5M	14.01	49.00	14.4M
F-Graph	42.50	73.92	19.6M	14.48	50.14	10.6M
S-Graph	43.37	74.43	19.6M	15.15	50.22	10.6M
FE-Graph	43.07	74.60	21.3M	14.84	51.53	12.7M
SE-Graph	44.33*	75.49*	21.3M	16.22*	52.13*	12.2M
SEK-Graph	**44.72***	**75.74***	21.8M	**17.17***	**53.16***	12.7M

Table 1: Main results on the sentence ordering task (The top half show results on NIPS Abstract and ANN Abstract, while the bottom half lists results on arXiv Abstract and SIND). #pm is the number of parameters, † indicates previously reported scores and * means significant at $p<0.01$ over F-Graph on each test set. Here we conduct 1,000 bootstrap tests (Koehn, 2004) to measure the significance in the metric score differences.

Model	arXiv Abstract		SIND	
	head	tail	head	tail
LSTM+PtrNet †	90.47	66.49	74.66	53.30
ATTOrderNet †	91.00	68.08	76.00	54.42
F-Graph	91.43	68.56	76.53	56.02
S-Graph	91.99	69.74	77.07	56.28
SE-Graph	92.28	70.45	78.12	56.68
SEK-Graph	92.23	70.46	79.01	57.27

Table 2: The ratio of correctly predicting first and last sentences on arXiv Abstract and SIND. † indicates previously reported scores.

Accuracies on Predicting the First and Last Sentences. Due to the crucial absolute positions of both the first and last sentences in a paragraph, we follow Cui et al. (2018) to

conduct experiments on predicting them. Experimental results on arXiv Abstract and SIND are listed in Table 2. We can observe that SE-Graph and SEK-Graph significantly surpass ATTOrderNet. Again, all these results echo previous experimental results and witness the advantages of our models.

5.4 Ablation Study

Model	S-Graph		SE-Graph		SEK-Graph	
	Acc	PMR	Acc	PMR	Acc	PMR
Original	58.06	43.37	58.91	44.33	59.15	44.72
Shuffle edges	57.06	42.41	57.46	42.84	57.99	43.53
Remove edge labels	—	—	58.51	43.96	58.52	43.99
Remove 10% entities	57.79	43.26	58.67	44.17	58.60	43.93
Remove 50% entities	57.57	42.83	57.84	43.18	58.16	43.68
Remove priori	—	—	—	—	58.91	44.46
Share parameters	—	—	55.30	40.31	58.31	43.65

Table 3: Ablation study of our graph structure on arXiv Abstract, where *Share Parameters* means employing the same parameters to update entity and sentence nodes.

Finally, we conducted ablation studies on arXiv Abstract, which is the largest one among our datasets, to inspect the impacts of various elements (entities and edges) on our model. Here we choose S-Graph, SE-Graph and SEK-Graph as experimental models, because both of them leverage entity information to construct graph representations. Table 3 reports the experimental results, where we can further obtain the following observations:

First, the utilizations of entities are indeed able to reduce noise in graph representations. Specifically, when shuffling edges to introduce noise into the graph representations, we find that the performances of these three models degrade significantly. Nevertheless, the resulting PMR of S-Graph (42.41 as shown in Line 4 of Table 3) is still comparable with the PMR of F-Graph (42.50 as shown in Line 14 of Table 1). This result indicates that fully-connected graphs are also noisy, especially because F-Graph has the same number of parameters as S-Graph.

Second, we also find that compared with removing or shuffling edges, removing edge labels leads to less performance drops. The underlying reason is the labels indeed provide useful information for our graph encoder. Besides, our graph encoders is also able to automatically learn some labels.

Third, since our entities are derived from the parsing results, removing entities is essentially a way to introduce syntactic parsing noise. When we only remove 10% entities, S-Graph, SE-Graph and SEK-Graph have slight performance decreases. Further, randomly removing 50% entities leads to significant performance drops. Note that in this case, the performance of SE-Graph is slightly worse than original model of S-Graph, demonstrating that the improvement of SE-Graph over S-Graph is not only derived from simply introducing more parameters. Overall, these results not only demonstrate the robustness of our

Input Sentences	(1) My best *friend* and i went out to meet with some other *friends*. (2) A *man* was shocked to see us after so many years. (3) The *man* introduced us to his new *wife*. (4) We went back to another *place* so continue catching up. (5) We took a group of *pictures* to remember this *event* for all *times*.
Ground Truth	(1) (2) (3) (4) (5)
F-Graph	(1) (3) (2) (4) (5)
S-Graph	(1) (2) (4) (3) (5)
SE-Graph	(1) (2) (3) (4) (5)
SEK-Graph	(1) (2) (3) (4) (5)
Input Sentences	(1) *Everyone* picked the *player* they thought would do the best. (2) *Mascots* meet before the *game*. (3) The *pitcher* was voted the *number one*. (4) Then he stopped and changed his *mind*. (5) He got ready to throw the *ball*.
Ground Truth	(2) (1) (3) (5) (4)
F-Graph	(1) (5) (3) (4) (2)
S-Graph	(1) (5) (3) (4) (2)
SE-Graph	(1) (5) (3) (4) (2)
SEK-Graph	(2) (1) (3) (5) (4)
Input Sentences	(1) The local parish holds a *craft* show each *year*. (2) *Lots* of *folks* come out and set up *tables* to sell their *crafts*. (3) Some of these *crafts* are very unique and take a lot of *talent* to make. (4) *Folks* of all *ages* come out to peruse the *crafts* for *sale*. (5) Some of the crafters even dress up in unique *costumes* as part of their *selling act*.
Ground Truth	(1) (2) (3) (4) (5)
F-Graph	(1) (2) (4) (3) (5)
S-Graph	(1) (2) (3) (5) (4)
SE-Graph	(1) (4) (2) (5) (3)
SEK-Graph	(4) (1) (3) (5) (2)

Table 4: Sentence ordering results produced by different models. Italic text denotes entities.

models against potential parsing accuracy drops on certain domains, such as medical and chemistry, but also indicate the importance of introducing entities.

Fourth, removeing the priori in SEK-Graph leads to a performance drop, showing that the word-level relatedness from external knowledge base can help the model to better capture the semantic relatedness between entities.

Finally, we observe that sharing parameters makes the final performance drop significantly. This result is reasonable since entity nodes play significantly different roles from sentence nodes. Therefore, modeling them separately is a better choice for our encoders.

5.5 Case Study

Table 4 shows two examples to illustrate the superiority of our models as well as an example for error analysis. In the first example, both SE-Graph and SEK-Graph are able to produce

the correct sequence of ordered sentences, since they fully exploit the co-occurrence between sentences and entities. In the second example, only SEK-Graph can make a totally correct prediction. This is because the entities *game* and *player*, *pitcher* and *ball* help SEK-Graph make a totally correct prediction. In the third example, the SE-Graph and SEK-Graph containing entity nodes perform worse than F-Graph and S-Graph, and the SEK-Graph makes an even worse prediction. The SE-Graph may excessively rely on the entity *folks*, while the SEK-Graph suffers from errors from the parser. Concretely, the parser tags *lot* and *lots* as nouns and then the Wordnet toolkit relates them to *selling*, misleading the model severely.

5.6 Summary Coherence Evaluation

As mentioned in previous work (Barzilay & Lapata, 2005, 2008), sentence ordering models can be used to refine arranging more coherent text for several downstream tasks, such as extractive multi-document summarization (Bollegala et al., 2006; Nayeem & Chali, 2017). Here we follow Nayeem and Chali (2017) to evaluate our sentence ordering models on the multi-document summarization task DUC 2004 (Task-2). However, since the DUC dataset is too small to effectively train neural sentence ordering models, we first pretrain our models on a large-scale summarization dataset (Fabbri et al., 2019), where the numbers of summaries in the training and validation sets are 44,972, 5,622, respectively, and the average number of sentences in each summary is 9.97. Then, we apply LexRank (Erkan & Radev, 2004) on DUC 2004 (Task-2) to extract summaries and utilize pretrained models to reorder the extracted summaries. It should be noted that ROUGE scores (Lin & Hovy, 2003) only focus on the content similarity between extracted summaries and references but are insensitive to coherence (Lapata & Barzilay, 2005; Nayeem & Chali, 2017). Therefore, we adopt the coherence probability proposed by Nayeem and Chali (2017) to evaluate the coherence of the summaries reordered by different models. Formally, the coherence probability of a summary x is defined as follows:

$$\text{Coherence}(x) = \frac{\sum_{i=1}^{n-1} \text{Sim}(x_i, x_{i+1})}{n-1}, \tag{30}$$

$$\text{Sim}(x_i, x_{i+1}) = \lambda * \text{NESim}(x_i, x_{i+1}) + (1-\lambda) * \text{CosSim}(x_i, x_{i+1}), \tag{31}$$

$$\text{NESim}(x_i, x_{i+1}) = \frac{|\text{NE}(x_i) \bigcap \text{NE}(x_{i+1})|}{min(|\text{NE}(x_i)|, |\text{NE}(x_{i+1})|)}, \tag{32}$$

where n is the number of sentences in x, $\text{NE}(x_*)$ denotes all named entities in the sentence x_*, and $\text{CosSim}(x_i, x_{i+1})$ calculates the cosine similarity between sentence embeddings of x_i and x_{i+1}. Here, we directly define the sentence embedding of a sentence as the weighted sum of its word embeddings. Besides, we choose $\lambda = 0.8$, giving more preference to the named entities.

The results are displayed in Table 5. We can observe that compared with other models, the reordered summaries by SEK-Graph achieve the higher coherence scores, verifying the effectiveness of our proposed model in the downstream task.

Model	Coherence
LexRank	45.68
F-Graph	47.56
S-Graph	49.13
SE-Graph	51.17
SEK-Graph	51.46

Table 5: Coherence probabilities of reordered summaries.

6. Conclusion

In this paper, we have presented a novel external knowledge enhanced graph-based neural network for sentence ordering. Concretely, we first propose two kinds of graph representations to model both the semantic relevance between coherent sentences and the co-occurrence between sentences and entities. Then, based on these graph representations, we introduce GRN to encode input sentences by performing semantic transitions among connected nodes. Compared with previous models, ours not only is able to reduce the noise brought by relationship modeling between incoherent sentences, but also can fully leverage entity information for paragraph encoding. Experimental results and in-depth analysis on several benchmark datasets prove the superiority of our model over the state-of-the-art and other baselines.

In the future, we plan to design a fault-tolerant architecture to mitigating error propagation from external tools. Besides, incorporating more powerful decoding algorithms (Zhang et al., 2018a; Yin et al., 2020a) is a promising direction.

Acknowledgments

The project was supported by National Natural Science Foundation of China (No. 62036004, No. 61672440), the Fundamental Research Funds for the Central Universities (Grant No. ZK1024), and Natural Science Foundation of Fujian Province of China (No.2020J06001).

Yongjing Yin and Shaopeng Lai contribute equally and Jinsong Su is the corresponding author.

References

Barzilay, R., Elhadad, N., & McKeown, K. R. (2002). Inferring strategies for sentence ordering in multidocument news summarization. In *JAIR*, *17*, 35–55.

Barzilay, R., & Lapata, M. (2005). Modeling local coherence: An entity-based approach. In *ACL*, pp. 141–148.

Barzilay, R., & Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Comput. Linguistics* , *34*(1), 1–34.

Barzilay, R., & Lee, L. (2004). Catching the drift: Probabilistic content models, with applications to generation and summarization. In *NAACL*, pp. 113–120.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., & Sima'an, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. In *EMNLP*, pp. 1957–1967.

Beck, D., Haffari, G., & Cohn, T. (2018). Graph-to-sequence learning using gated graph neural networks. In *ACL*, pp. 273–283.

Bollegala, D., Okazaki, N., & Ishizuka, M. (2006). A bottom-up approach to sentence ordering for multi-document summarization. In *ACL*, pp. 385–392.

Burstein, J., Tetreault, J. R., & Andreyev, S. (2010). Using entity-based features to model coherence in student essays. In *NAACL*, pp. 681–684.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, pp. 1724–1734.

Cui, B., Li, Y., Chen, M., & Zhang, Z. (2018). Deep attentive sentence ordering network. In *EMNLP*, pp. 4340–4349.

Elsner, M., & Charniak, E. (2011). Extending the entity grid with entity-specific features. In *ACL*, pp. 125–129.

Erkan, G., & Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, *22*, 457–479.

Fabbri, A. R., Li, I., She, T., Li, S., & Radev, D. R. (2019). Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *ACL*, pp. 1074–1084.

Galanis, D., Lampouras, G., & Androutsopoulos, I. (2012). Extractive multi-document summarization with integer linear programming and support vector regression. In *COLING*, pp. 911–926.

Gong, J., Chen, X., Qiu, X., & Huang, X. (2016). End-to-end neural sentence ordering using pointer network. *CoRR*, *abs/1611.04953*.

Guinaudeau, C., & Strube, M. (2013). Graph-based local coherence modeling. In *ACL*, pp. 93–103.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Holtzman, A., Buys, J., Forbes, M., Bosselut, A., Golub, D., & Choi, Y. (2018). Learning to write with cooperative discriminators. In *ACL*, pp. 1638–1649.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *EMNLP*, pp. 388–395.

Konstas, I., & Lapata, M. (2012). Unsupervised concept-to-text generation with hypergraphs. In *NAACL*, pp. 752–761.

Lapata, M. (2003). Probabilistic text structuring: Experiments with sentence ordering. In *ACL*, pp. 545–552.

Lapata, M., & Barzilay, R. (2005). Automatic evaluation of text coherence: Models and representations. In *IJCAI*, pp. 1085-1090.

Li, J., & Hovy, E. H. (2014). A model of coherence based on distributed sentence representation. In *EMNLP*, pp. 2039–2048.

Li, J., & Jurafsky, D. (2017). Neural net models of open-domain discourse coherence. In *EMNLP*, pp. 198-209.

Li, W., Xu, J., He, Y., Yan, S., Wu, Y., & Sun, X. (2019). Coherent comments generation for chinese articles with a graph-to-sequence model. In *ACL*, pp. 4843–4852.

Lin, C., & Hovy, E. H. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*, pp. 150–157.

Logeswaran, L., Lee, H., & Radev, D. R. (2018). Sentence ordering and coherence modeling using recurrent neural networks. In *AAAI*, pp. 5285–5292.

Marcheggiani, D., & Titov, I. (2017a). Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*, pp. 1506–1515.

Marcheggiani, D., & Titov, I. (2017b). Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*, pp. 1506–1515.

Mohiuddin, T., Joty, S. R., & Nguyen, D. T. (2018). Coherence modeling of asynchronous conversations: A neural entity grid approach. In *ACL*, pp. 558–568.

Moon, H. C., Mohiuddin, T., Joty, S. R., & Chi, X. (2019). A unified neural coherence model. In *EMNLP-IJCNLP*, pp. 2262–2272.

Nallapati, R., Zhai, F., & Zhou, B. (2012). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pp. 3075–3081.

Nayeem, M. T., & Chali, Y. (2017). Extract with order for coherent multi-document summarization. In *ACL Workshop*, pp. 51–56.

Nguyen, D. T., & Joty, S. R. (2017). A neural local coherence model. In *ACL*, pp. 1320-1330.

Radev, D. R., Joseph, M. T., Gibson, B. R., & Muthukrishnan, P. (2016). A bibliometric and network analysis of the field of computational linguistics. *JASIST*, *67*(3), 683–706.

Sahu, S. K., Christopoulou, F., Miwa, M., & Ananiadou, S. (2019). Inter-sentence relation extraction with document-level graph convolutional neural network. In *ACL*, pp. 4309–4316.

Song, L., Gildea, D., Zhang, Y., Wang, Z., & Su, J. (2019). Semantic neural machine translation using amr. *CoRR*, *abs/1902.07282*.

Song, L., Zhang, Y., Wang, Z., & Gildea, D. (2018a). A graph-to-sequence model for amr-to-text generation. In *ACL*, pp. 1616–1626.

Song, L., Zhang, Y., Wang, Z., & Gildea, D. (2018b). N-ary relation extraction using graph-state lstm. In *EMNLP*, pp. 2226–2235.

Vashishth, S., Bhandari, M., Yadav, P., Rai, P., Bhattacharyya, C., & Talukdar, P. P. (2019). Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *ACL*, pp. 3308–3318.

Vashishth, S., Dasgupta, S. S., Ray, S. N., & Talukdar, P. (2018). Dating documents using graph convolution networks. In *ACL*, pp. 1605–1615.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *NIPS*, pp. 6000–6010.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *NIPS*, pp. 2692–2700.

Wang, Z., Lv, Q., Lan, X., & Zhang, Y. (2018). Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, pp. 349–357.

Yin, Y., Meng, F., Su, J., Ge, Y., Song, L., Zhou, J., & Luo, J. (2020a). Enhancing pointer network for sentence ordering with pairwise ordering predictions. In *AAAI*, pp. 9482–9489.

Yin, Y., Meng, F., Su, J., Zhou, C., Yang, Z., Zhou, J., & Luo, J. (2020b). A novel graph-based multi-modal fusion encoder for neural machine translation. In *ACL*, pp. 3025–3035.

Yin, Y., Song, L., Su, J., Zeng, J., Zhou, C., & Luo, J. (2019). Graph-based neural sentence ordering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 5387–5393.

Yu, A. W., Dohan, D., Luong, M., Zhao, R., Chen, K., Norouzi, M., & Le, Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, *abs/1212.5701*.

Zeng, X., Li, J., Wang, L., Beauchamp, N., Shugars, S., & Wong, K. (2018). Microblog conversation recommendation via joint modeling of topics and discourse. In *NAACL-HLT*, pp. 375–385.

Zhang, X., Su, J., Qin, Y., Liu, Y., Ji, R., & Wang, H. (2018a). Asynchronous bidirectional decoding for neural machine translation. In *AAAI*, pp. 5698–5705.

Zhang, Y., Liu, Q., & Song, L. (2018b). Sentence-state lstm for text representation. In *ACL*, pp. 317–327.

Zhang, Y., Qi, P., & Manning, C. D. (2018c). Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*, pp. 2205–2215.