Story Ending Generation with Incremental Encoding and Commonsense Knowledge

Jian Guan[2*] , **Yansen Wang**[1*], **Minlie Huang**[1†]
[1]Dept. of Computer Science & Technology, Tsinghua University, Beijing 100084, China
[1]Institute for Artificial IntelligenceTsinghua University (THUAI), China
[1]Beijing National Research Center for Information Science and Technology, China
[2]Dept. of Physics, Tsinghua University, Beijing 100084, China
guanj15@mails.tsinghua.edu.cn;ys-wang15@mails.tsinghua.edu.cn;
aihuang@tsinghua.edu.cn

arXiv:1808.10113v2 [cs.CL] 10 Nov 2018

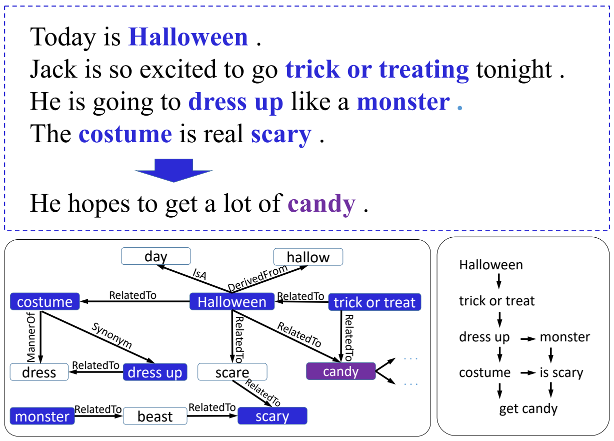
Abstract

Generating a reasonable ending for a given story context, i.e., story ending generation, is a strong indication of story comprehension. This task requires not only to understand the context clues which play an important role in planning the plot, but also to handle implicit knowledge to make a reasonable, coherent story. In this paper, we devise a novel model for story ending generation. The model adopts an incremental encoding scheme to represent context clues which are spanning in the story context. In addition, commonsense knowledge is applied through multi-source attention to facilitate story comprehension, and thus to help generate coherent and reasonable endings. Through building context clues and using implicit knowledge, the model is able to produce reasonable story endings. Automatic and manual evaluation shows that our model can generate more reasonable story endings than state-of-the-art baselines. [1]

Introduction

Story generation is an important but challenging task because it requires to deal with logic and implicit knowledge (Li et al. 2013; Soo, Lee, and Chen 2016; Ji et al. 2017; Jain et al. 2017; Martin et al. 2018; Clark, Ji, and Smith 2018). Story ending generation aims at concluding a story and completing the plot given a story context. We argue that solving this task involves addressing the following issues: 1) Representing the **context clues** which contain key information for planning a reasonable ending; and 2) Using **implicit knowledge** (e.g., commonsense knowledge) to facilitate understanding of the story and better predict what will happen next.

Comparing to textual entailment or reading comprehension (Dagan, Glickman, and Magnini 2006; Hermann et al. 2015) story ending generation requires more to deal with the logic and causality information that may span multiple sentences in a story context. The logic information in story can be captured by the appropriate sequence of events[2] or entities occurring in a sequence of sentences, and the chronological order or causal relationship between events or entities. The ending should be generated from the whole context clue rather than merely inferred from a single entity or the last sentence. It is thus important for story ending generation to represent the context clues for predicting what will happen in an ending.


Today is **Halloween** .
Jack is so excited to go **trick or treating** tonight .
He is going to **dress up** like a **monster** .
The **costume** is real **scary** .

He hopes to get a lot of **candy** .

Figure 1: A story example. Words in blue/purple are events and entities. The bottom-left graph is retrieved from ConceptNet and the bottom-right graph represents how events and entities form the context clue.

However, deciding a reasonable ending not only depends on representing the context clues properly, but also on the ability of language understanding with implicit knowledge that is beyond the text surface. Humans use their own experiences and implicit knowledge to help understand a story. As shown in Figure 1, the ending talks about *candy* which can be viewed as commonsense knowledge about *Halloween*. Such knowledge can be crucial for story ending generation.

Figure 1 shows an example of a typical story in the ROCStories corpus (Mostafazadeh et al. 2016b). In this example, the events or entities in the story context constitute the

[*] Authors contributed equally to this work.
[†] Corresponding author: Minlie Huang.

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]Our codes and data are available at `https://github.com/JianGuanTHU/StoryEndGen`.

[2]Event in this paper refers to a verb or simple action such as *dress up, trick or treating* as shown in Figure 1.

context clues which reveal the logical or causal relationships between events or entities. These concepts, including *Halloween*, *trick or treat*, and *monster*, are connected as a graph structure. A reasonable ending should consider all the connected concepts rather than just some individual one. Furthermore, with the help of commonsense knowledge retrieved from ConceptNet (Speer and Havasi 2012), it is easier to infer a reasonable ending with the knowledge that *candy* is highly related to *Halloween*.

To address the two issues in story ending generation, we devise a model that is equipped with an **incremental encoding scheme** to encode context clues effectively, and a **multi-source attention mechanism** to use *commonsense knowledge*. The representation of the context clues is built through incremental reading (or encoding) of the sentences in the story context one by one. When encoding a current sentence in a story context, the model can attend not only to the words in the preceding sentence, but also the knowledge graphs which are retrieved from ConceptNet for each word. In this manner, commonsense knowledge can be encoded in the model through graph representation techniques, and therefore, be used to facilitate understanding story context and inferring coherent endings. Integrating the context clues and commonsense knowledge, the model can generate more reasonable endings than state-of-the-art baselines.

Our contributions are as follows:

- To assess the machinery ability of story comprehension, we investigate story ending generation from two new angles. One is to modeling logic information via incremental context clues and the other is the utility of implicit knowledge in this task.
- We propose a neural model which represents context clues by incremental encoding, and leverages commonsense knowledge by multi-source attention, to generate logical and reasonable story endings. Results show that the two techniques are effective in capturing the coherence and logic of story.

Related Work

The corpus we used in this paper was first designed for Story Cloze Test (SCT) (Mostafazadeh et al. 2016a), which requires to select a correct ending from two candidates given a story context. Feature-based (Chaturvedi, Peng, and Dan 2017; Lin, Sun, and Han 2017; Mostafazadeh et al. 2016b) or neural (Mostafazadeh et al. 2016b; Wang, Liu, and Zhao 2017; Cai, Tu, and Gimpel 2017) classification models are proposed to measure the coherence between a candidate ending and a story context from various aspects such as event, sentiment, and topic. However, story ending generation (Li, Ding, and Liu 2018; Zhao et al. 2018) is more challenging in that the task requires to modeling context clues and implicit knowledge to produce reasonable endings.

Story generation, moving forward to complete story comprehension, is approached as selecting a sequence of events to form a story by satisfying a set of criteria (Li et al. 2013). Previous studies can be roughly categorized into two lines: rule-based methods and neural models. Most of the traditional rule-based methods for story generation (Li et al. 2013; Soo, Lee, and Chen 2016) retrieve events from a knowledge base with some pre-specified semantic relations.

Neural models for story generation has been widely studied with sequence-to-sequence (seq2seq) learning (Roemmele 2016). And various contents such as photos and independent descriptions are largely used to inspire the story (Jain et al. 2017).To capture the deep meaning of key entities and events, Ji et al. (2017) and Clark, Ji, and Smith (2018) explicitly modeled the entities mentioned in story with dynamic representation, and Martin et al. (2018) decomposed the problem into planning successive events and generating sentences from some given events. Fan, Lewis, and Dauphin (2018) adopted a hierarchical architecture to generate the whole story from some given keywords.

Commonsense knowledge is beneficial for many natural language tasks such as semantic reasoning and text entailment, which is particularly important for story generation. LoBue and Yates (2011) characterized the types of commonsense knowledge mostly involved in recognizing textual entailment. Afterwards, commonsense knowledge was used in natural language inference (R. Bowman et al. 2015; Zhang et al. 2017) and language generation (Zhou et al. 2018). Mihaylov and Frank (2018) incorporated external commonsense knowledge into a neural cloze-style reading comprehension model. Rashkin et al. (2018) performed commonsense inference on people's intents and reactions of the event's participants given a short text. Similarly, Knight et al. (2018) introduced a new annotation framework to explain psychology of story characters with commonsense knowledge. And commonsense knowledge has also been shown useful to choose a correct story ending from two candidate endings (Lin, Sun, and Han 2017; Li et al. 2018).

Methodology

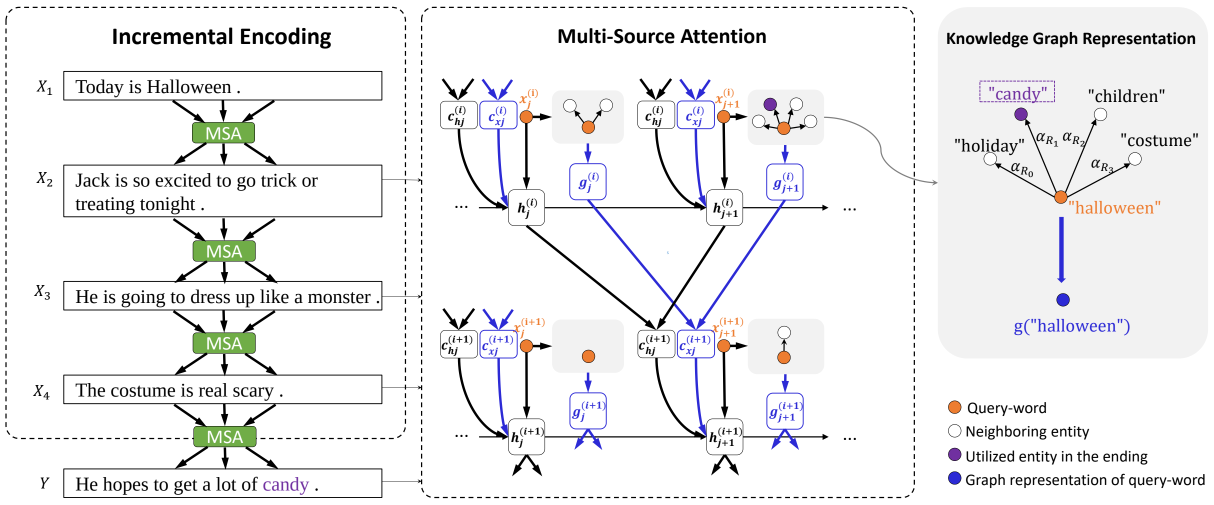
Overview

The task of story ending generation can be stated as follows: given a story context consisting of a sentence sequence $X = \{X_1, X_2, \cdots, X_K\}$[3], where $X_i = x_1^{(i)} x_2^{(i)} \cdots x_{l_i}^{(i)}$ represents the i-th sentence containing l_i words, the model should generate a one-sentence ending $Y = y_1 y_2 ... y_l$ which is reasonable in logic, formally as

$$Y^* = \underset{Y}{argmax}\, \mathcal{P}(Y|X). \tag{1}$$

As aforementioned, context clue and commonsense knowledge is important for modeling the logic and casual information in story ending generation. To this end, we devise an **incremental encoding scheme** based on the general encoder-decoder framework (Sutskever, Vinyals, and Le 2014). As shown in Figure 2, the scheme encodes the sentences in a story context incrementally with a **multi-source attention mechanism**: when encoding sentence X_i, the encoder obtains a context vector which is an attentive read of the hidden states, and the graph vectors of the preceding sentence X_{i-1}. In this manner, the relationship between words

[3]Adjacent sentences in story context have much logic connection, temporal dependency, and casual relationship.



Incremental Encoding
X_1 Today is Halloween .
MSA
X_2 Jack is so excited to go trick or treating tonight .
MSA
X_3 He is going to dress up like a monster .
MSA
X_4 The costume is real scary .
MSA
Y He hopes to get a lot of candy .
Multi-Source Attention
Knowledge Graph Representation
"candy"
"children"
"holiday"
"costume"
"halloween"
g("halloween")
Query-word
Neighboring entity
Utilized entity in the ending
Graph representation of query-word

Figure 2: Model overview. The model is equipped with incremental encoding (IE) and multi-source attention (MSA). $x_j^{(i)}$: the j-th word in sentence i; $\mathbf{c}_{hj}^{(i)}$: state context vector; $\mathbf{c}_{xj}^{(i)}$: knowledge context vector; $\mathbf{g}_j^{(i)}$: graph vector of word $x_j^{(i)}$; $\mathbf{h}_j^{(i)}$: j-th hidden state of sentence i. The state (knowledge) context vectors are attentive read of hidden states (graph vectors) in the preceding sentence.

(some are entities or events) in sentence X_{i-1} and those in X_i is built *incrementally*, and therefore, the chronological order or causal relationship between entities (or events) in adjacent sentences can be captured implicitly. To leverage commonsense knowledge which is important for generating a reasonable ending, a one-hop knowledge graph for each word in a sentence is retrieved from ConceptNet, and each graph can be represented by a vector in two ways. The incremental encoder not only attends to the hidden states of X_{i-1}, but also to the graph vectors at each position of X_{i-1}. By this means, our model can generate more reasonable endings by representing context clues and encoding commonsense knowledge.

Background: Encoder-Decoder Framework

The encoder-decoder framework is a general framework widely used in text generation. Formally, the model encodes the input sequence $X = x_1x_2 \cdots x_m$ into a sequence of hidden states, as follows,

$$\mathbf{h}_t = \mathbf{LSTM}(\mathbf{h}_{t-1}, \boldsymbol{e}(x_t)), \tag{2}$$

where $\mathbf{h}_t$ denotes the hidden state at step t and $\boldsymbol{e}(x)$ is the word vector of x.

At each decoding position, the framework will generate a word by sampling from the word distribution $\mathcal{P}(y_t|y_{<t}, X)$ ($y_{<t} = y_1y_2 \cdots y_{t-1}$ denotes the sequences that has been generated before step t), which is computed as follows:

$$\mathcal{P}(y_t|y_{<t}, X) = \mathbf{softmax}(\mathbf{W}_0\boldsymbol{s}_t + \mathbf{b}_0), \tag{3}$$

$$\mathbf{s}_t = \mathbf{LSTM}(\mathbf{s}_{t-1}, \boldsymbol{e}(y_{t-1}), \mathbf{c}_{t-1}), \tag{4}$$

where $\mathbf{s}_t$ denotes the decoder state at step t. When an attention mechanism is applied, $\mathbf{c}_{t-1}$ is an attentive read of the context, which is a weighted sum of the encoder's hidden states as $\mathbf{c}_{t-1} = \sum_{i=1}^{m} \alpha_{(t-1)i}\mathbf{h}_i$, and $\alpha_{(t-1)i}$ measures the association between the decoder state $\mathbf{s}_{t-1}$ and the encoder state $\mathbf{h}_i$. Refer to (Bahdanau, Cho, and Bengio 2014) for more details.

Incremental Encoding Scheme

Straightforward solutions for encoding the story context can be: 1) Concatenating the K sentences to a long sentence and encoding it with an LSTM ; or 2) Using a hierarchical LSTM with hierarchical attention (Yang et al. 2016), which firstly attends to the hidden states of a sentence-level LSTM, and then to the states of a word-level LSTM. However, these solutions are not effective to represent the context clues which may capture the key logic information. Such information revealed by the chronological order or causal relationship between events or entities in adjacent sentences.

To better represent the context clues, we propose an incremental encoding scheme: when encoding the current sentence X_i, it obtains a context vector which is an attentive read of the preceding sentence X_{i-1}. In this manner, the order/relationship between the words in adjacent sentences can be captured implicitly.

This process can be stated formally as follows:

$$\mathbf{h}_j^{(i)} = \mathbf{LSTM}(\mathbf{h}_{j-1}^{(i)}, \boldsymbol{e}(x_j^{(i)}), \mathbf{c}_{\mathbf{l}j}^{(i)}),\ i \geq 2. \tag{5}$$

where $\mathbf{h}_j^{(i)}$ denotes the hidden state at the j-th position of the i-th sentence, $\boldsymbol{e}(x_j^{(i)})$ denotes the word vector of the j-th word $x_j^{(i)}$. $\mathbf{c}_{\mathbf{l},j}^{(i)}$ is the context vector which is an attentive read of the *preceding* sentence X_{i-1}, conditioned on $\mathbf{h}_{j-1}^{(i)}$. We will describe the context vector in the next section.

During the decoding process, the decoder obtains a context vector from the last sentence X_K in the context to utilize the context clues. The hidden state is obtained as below:

$$\mathbf{s}_t = \mathbf{LSTM}(\mathbf{s}_{t-1}, e(y_{t-1}), \mathbf{c_l}_t), \tag{6}$$

where $\mathbf{c_l}_t$ is the context vector which is the attentive read of the last sentence X_K, conditioned on $\mathbf{s}_{t-1}$. More details of the context vector will be presented in the next section.

Multi-Source Attention (MSA)

The context vector ($\mathbf{c_l}$) plays a key role in representing the context clues because it captures the relationship between words (or states) in the current sentence and those in the preceding sentence. As aforementioned, story comprehension sometime requires the access of implicit knowledge that is beyond the text. Therefore, the context vector consists of two parts, computed with multi-source attention. The first one $\mathbf{c}_{\mathbf{h}j}^{(i)}$ is derived by attending to the hidden states of the preceding sentence, and the second one $\mathbf{c}_{\mathbf{x}j}^{(i)}$ by attending to the knowledge graph vectors which represent the one-hop graphs in the preceding sentence. The MSA context vector is computed as follows:

$$\mathbf{c}_{\mathbf{l}j}^{(i)} = \mathbf{W_l}([\mathbf{c}_{\mathbf{h}j}^{(i)}; \mathbf{c}_{\mathbf{x}j}^{(i)}]) + \mathbf{b_l}, \tag{7}$$

where $\oplus$ indicates vector concatenation. Hereafter, $\mathbf{c}_{\mathbf{h}j}^{(i)}$ is called *state context vector*, and $\mathbf{c}_{\mathbf{x}j}^{(i)}$ is called *knowledge context vector*.

The **state context vector** is a weighted sum of the hidden states of the preceding sentence X_{i-1} and can be computed as follows:

$$\mathbf{c}_{\mathbf{h}j}^{(i)} = \sum_{k=1}^{l_{i-1}} \alpha_{h_k,j}^{(i)} \mathbf{h}_k^{(i-1)}, \tag{8}$$

$$\alpha_{h_k,j}^{(i)} = \frac{e^{\beta_{h_k,j}^{(i)}}}{\sum\limits_{m=1}^{l_{i-1}} e^{\beta_{h_m,j}^{(i)}}}, \tag{9}$$

$$\beta_{h_k,j}^{(i)} = \mathbf{h}_{j-1}^{(i)\mathrm{T}} \mathbf{W_s} \mathbf{h}_k^{(i-1)}, \tag{10}$$

where $\beta_{h_k,j}^{(i)}$ can be viewed as a weight between hidden state $\mathbf{h}_{j-1}^{(i)}$ in sentence X_i and hidden state $\mathbf{h}_k^{(i-1)}$ in the preceding sentence X_{i-1}.

Similarly, the **knowledge context vector** is a weighted sum of the **graph vectors** for the preceding sentence. Each word in a sentence will be used as a query to retrieve a one-hop commonsense knowledge graph from ConceptNet, and then, each graph will be represented by a **graph vector**. After obtaining the graph vectors, the knowledge context vector can be computed by:

$$\mathbf{c}_{\mathbf{x}j}^{(i)} = \sum_{k=1}^{l_{i-1}} \alpha_{x_k,j}^{(i)} \mathbf{g}(x_k^{(i-1)}), \tag{11}$$

$$\alpha_{x_k,j}^{(i)} = \frac{e^{\beta_{x_k,j}^{(i)}}}{\sum\limits_{m=1}^{l_{i-1}} e^{\beta_{x_m,j}^{(i)}}}, \tag{12}$$

$$\beta_{x_k,j}^{(i)} = \mathbf{h}_{j-1}^{(i)\mathrm{T}} \mathbf{W_k} \mathbf{g}(x_k^{(i-1)}), \tag{13}$$

where $\mathbf{g}(x_k^{(i-1)})$ is the **graph vector** for the graph which is retrieved for word $x_k^{(i-1)}$. Different from $\boldsymbol{e}(x_k^{(i-1)})$ which is the word vector, $\mathbf{g}(x_k^{(i-1)})$ encodes commonsense knowledge and extends the semantic representation of a word through neighboring entities and relations.

During the decoding process, the knowledge context vectors are similarly computed by attending to the last input sentence X_K. There is no need to attend to all the context sentences because the context clues have been propagated within the incremental encoding scheme.

Knowledge Graph Representation

Commonsense knowledge can facilitate language understanding and generation. To retrieve commonsense knowledge for story comprehension, we resort to ConceptNet[4] (Speer and Havasi 2012). ConceptNet is a semantic network which consists of triples $R = (h, r, t)$ meaning that head concept h has the relation r with tail concept t. Each word in a sentence is used as a query to retrieve a one-hop graph from ConceptNet. The knowledge graph for a word extends (encodes) its meaning by representing the graph from neighboring concepts and relations.

There have been a few approaches to represent commonsense knowledge. Since our focus in this paper is on using knowledge to benefit story ending generation, instead of devising new methods for representing knowledge, we adopt two existing methods: 1) **graph attention** (**?**; Zhou et al. 2018), and 2) **contextual attention** (Mihaylov and Frank 2018). We compared the two means of knowledge representation in the experiment.

Graph Attention Formally, the knowledge graph of word (or concept) x is represented by a set of triples, $\mathbf{G}(x) = \{R_1, R_2, \cdots, R_{N_x}\}$ (where each triple R_i has the same head concept x), and the graph vector $\boldsymbol{g}(x)$ for word x can be computed via *graph attention*, as below:

$$\mathbf{g}(x) = \sum_{i=1}^{N_x} \alpha_{R_i} [\mathbf{h}_i; \mathbf{t}_i], \tag{14}$$

$$\alpha_{R_i} = \frac{e^{\beta_{R_i}}}{\sum\limits_{j=1}^{N_x} e^{\beta_{R_j}}}, \tag{15}$$

$$\beta_{R_i} = (\mathbf{W_r}\mathbf{r}_i)^{\mathrm{T}}\, tanh(\mathbf{W_h}\mathbf{h}_i + \mathbf{W_t}\mathbf{t}_i), \tag{16}$$

where $(h_i, r_i, t_i) = R_i \in \mathbf{G}(x)$ is the i-th triple in the graph. We use word vectors to represent concepts, i.e. $\mathbf{h}_i = \boldsymbol{e}(h_i), \mathbf{t}_i = \boldsymbol{e}(t_i)$, and learn trainable vector $\mathbf{r}_i$ for relation r_i, which is randomly initialized.

Intuitively, the above formulation assumes that the knowledge meaning of a word can be represented by its neighboring concepts (and corresponding relations) in the knowledge base. Note that entities in ConceptNet are common words (such as *tree, leaf, animal*), we thus use word vectors to represent h/r/t directly, instead of using geometric embedding

[4]https://conceptnet.io

methods (e.g., TransE) to learn entity and relation embeddings. In this way, there is no need to bridge the representation gap between geometric embeddings and text-contextual embeddings (i.e., word vectors).

Contextual Attention When using *contextual attention*, the graph vector $\mathbf{g}(x)$ can be computed as follows:

$$\mathbf{g}(x) = \sum_{i=1}^{N_x} \alpha_{R_i} \mathbf{M}_{R_i}, \quad (17)$$

$$\mathbf{M}_{R_i} = BiGRU(\mathbf{h}_i, \mathbf{r}_i, \mathbf{t}_i), \quad (18)$$

$$\alpha_{R_i} = \frac{e^{\beta_{R_i}}}{\sum_{j=1}^{N_x} e^{\beta_{R_j}}}, \quad (19)$$

$$\beta_{R_i} = \mathbf{h}_{(x)}^{\mathrm{T}} \mathbf{W_c} \mathbf{M}_{R_i}, \quad (20)$$

where $\mathbf{M}_{R_i}$ is the final state of a *BiGRU* connecting the elements of triple R_i, which can be seen as the knowledge memory of the i-th triple, while $\mathbf{h}_{(x)}$ denotes the hidden state at the encoding position of word x.

Loss Function

As aforementioned, the incremental encoding scheme is central for story ending generation. To better model the chronological order and causal relationship between adjacent sentences, we impose supervision on the encoding network. At each encoding step, we also generate a distribution over the vocabulary, very similar to the decoding process:

$$\mathcal{P}(y_t|y_{<t}, X) = \mathbf{softmax}(\mathbf{W}_0 \mathbf{h}_j^{(i)} + \mathbf{b}_0), \quad (21)$$

Then, we calculate the negative data likelihood as loss function:

$$\Phi = \Phi_{en} + \Phi_{de} \quad (22)$$

$$\Phi_{en} = \sum_{i=2}^{K} \sum_{j=1}^{l_i} -\log \mathcal{P}(x_j^{(i)} = \widetilde{x}_j^{(i)} | x_{<j}^{(i)}, X_{<i}), \quad (23)$$

$$\Phi_{de} = \sum_t -\log \mathcal{P}(y_t = \tilde{y}_t | y_{<t}, X), \quad (24)$$

where $\widetilde{x}_j^{(i)}$ means the reference word used for encoding at the j-th position in sentence i, and $\tilde{y}_t$ represents the j-th word in the reference ending. Such an approach does not mean that at each step there is only one correct next sentence, exactly as many other generation tasks. Experiments show that it is better in logic than merely imposing supervision on the decoding network.

Experiments

Dataset

We evaluated our model on the ROCStories corpus (Mostafazadeh et al. 2016a). The corpus contains 98,162 five-sentence stories for evaluating story understanding and script learning. The original task is designed to select a correct story ending from two candidates, while our task is to generate a reasonable ending given a four-sentence story context. We randomly selected 90,000 stories for training and the left 8,162 for evaluation. The average number of words in $X_1/X_2/X_3/X_4/Y$ is 8.9/9.9/10.1/10.0/10.5 respectively. The training data contains 43,095 unique words, and 11,192 words appear more than 10 times. For each word, we retrieved a set of triples from ConceptNet and stored those whose head entity and tail entity are noun or verb, meanwhile both occurring in SCT. Moreover, we retained at most 10 triples if there are too many. The average number of triples for each query word is 3.4.

Baselines

We compared our models with the following state-of-the-art baselines:
Sequence to Sequence (Seq2Seq): A simple encoder-decoder model which concatenates four sentences to a long sentence with an attention mechanism (Luong, Pham, and Manning 2015).
Hierarchical LSTM (HLSTM): The story context is represented by a hierarchical LSTM: a word-level LSTM for each sentence and a sentence-level LSTM connecting the four sentences (Yang et al. 2016). A hierarchical attention mechanism is applied, which attends to the states of the two LSTMs respectively.
HLSTM+Copy: The copy mechanism (Gu et al. 2016) is applied to hierarchical states to copy the words in the story context for generation.
HLSTM+Graph Attention(GA): We applied multi-source attention HLSTM where commonsense knowledge is encoded by graph attention.
HLSTM+Contextual Attention(CA): Contextual attention is applied to represent commonsense knowledge.

Experiment Settings

The parameters are set as follows: GloVe.6B (Pennington, Socher, and Manning 2014) is used as word vectors, and the vocabulary size is set to 10,000 and the word vector dimension to 200. We applied 2-layer LSTM units with 512-dimension hidden states. These settings were applied to all the baselines.

The parameters of the LSTMs (Eq. 5 and 6) are shared by the encoder and the decoder.

Automatic Evaluation

We conducted the automatic evaluation on the 8,162 stories (the entire test set). We generated endings from all the models for each story context.

Evaluation Metrics We adopted *perplexity*(PPL) and *BLEU* (Papineni et al. 2002) to evaluate the generation performance. Smaller perplexity scores indicate better performance. *BLEU* evaluates n-gram overlap between a generated ending and a reference ending. However, since there is only one reference ending for each story context, BLEU scores will become extremely low for larger n. We thus experimented with $n = 1, 2$. Note also that there may exist multiple reasonable endings for the same story context.

Model	PPL	BLEU-1	BLEU-2	Gram.	Logic.
Seq2Seq	18.97	0.1864	0.0090	1.74	0.70
HLSTM	17.26	0.2459	0.0242	1.57	0.84
HLSTM+Copy	19.93	0.2469	0.0248	1.66	0.90
HLSTM+MSA(GA)	15.75	0.2588	0.0253	1.70	1.06
HLSTM+MSA(CA)	12.53	0.2514	0.0271	1.72	1.02
IE (ours)	11.04	0.2514	0.0263	**1.84**	1.10
IE+MSA(GA) (ours)	9.72	0.2566	0.0284	1.68	**1.26**
IE+MSA(CA) (ours)	**8.79**	**0.2682**	**0.0327**	1.66	1.24

Table 1: Automatic and manual evaluation results.

Results The results of the automatic evaluation are shown in Table 1, where IE means a simple incremental encoding framework that ablated the knowledge context vector from $\mathbf{c_l}$ in Eq. (7). Our models have lower perplexity and higher BLEU scores than the baselines. IE and IE+MSA have remarkably lower perplexity than other models. As for BLEU, IE+MSA(CA) obtained the highest BLEU-1 and BLEU-2 scores. This indicates that multi-source attention leads to generate story endings that have more overlaps with the reference endings.

Manual Evaluation

Manual evaluations are indispensable to evaluate the coherence and logic of generated endings. For manual evaluation, we randomly sampled 200 stories from the test set and obtained 1,600 endings from the eight models. Then, we resorted to Amazon Mechanical Turk (MTurk) for annotation. Each ending will be scored by three annotators and majority voting is used to select the final label.

Evaluation Metrics We defined two metrics - *grammar* and *logicality* for manual evaluation. Score 0/1/2 is applied to each metric during annotation.

Grammar (Gram.): Whether an ending is natural and fluent. Score 2 is for endings without any grammar errors, 1 for endings with a few errors but still understandable and 0 for endings with severe errors and incomprehensible.

Logicality (Logic.): Whether an ending is reasonable and coherent with the story context in logic. Score 2 is for reasonable endings that are coherent in logic, 1 for relevant endings but with some discrepancy between an ending and a given context, and 0 for totally incompatible endings.

Note that the two metrics are scored independently. To produce high-quality annotation, we prepared guidelines and typical examples for each metric score.

Results The results of the manual evaluation are also shown in Table 1. Note that the difference between IE and IE+MSA exists in that IE does not attend to knowledge graph vectors in a preceding sentence, and thus it does use any commonsense knowledge. The incremental encoding scheme without MSA obtained the best grammar score and our full mode IE+MSA(GA) has the best logicality score. All the models have fairly good grammar scores (maximum is 2.0), while the logicality scores differ remarkably, much lower than the maximum score, indicating the challenges of this task.

More specifically, **incremental encoding is effective** due to the facts: 1) IE is significantly better than Seq2Seq and HLSTM in grammar (Sign Test, 1.84 vs. 1.74/1.57, p-value=0.046/0.037, respectively), and in logicality (1.10 vs. 0.70/0.84, $\text{p-value}< 0.001/0.001$). 2) IE+MSA is significantly better than HLSTM+MSA in logicality (1.26 vs. 1.06, p-value=0.014 for GA; 1.24 vs. 1.02, p-value=0.022 for CA). This indicates that incremental encoding is more powerful than traditional (Seq2Seq) and hierarchical (HLSTM) encoding/attention in utilizing context clues. Furthermore, **using commonsense knowledge leads to significant improvements in logicality**. The comparison in logicality between IE+MSA and IE (1.26/1.24 vs. 1.10, p-value=0.028/0.042 for GA/CA, respectively), HLSTM+MSA and HLSTM (1.06/1.02 vs. 0.84, $\text{p-value}< 0.001/0.001$ for GA/CA, respectively), and HLSTM+MSA and HLSTM+Copy (1.06/1.02 vs. 0.90, p-value=0.044/0.048, respectively) all approve this claim. In addition, similar results between GA and CA show that commonsense knowledge is useful but multi-source attention is not sensitive to the knowledge representation scheme.

More detailed results are listed in Table 2. Comparing to other models, IE+MSA has a much larger proportion of endings that are good both in grammar and logicality (2-2). The proportion of good logicality (score=2.0) from IE+MSA is much larger than that from IE (45.0%+5.0%/41.0%+4.0% vs. 36.0%+2.0% for GA/CA, respectively), and also remarkable larger than those from other baselines. Further, HLSTM equipped with MSA is better than those without MSA, indicating that commonsense knowledge is helpful. And the kappa measuring inter-rater agreement is 0.29 for three annotators, which implies a fair agreement.

Gram.-Logic. Score	2-2	2-1	1-2	1-1
Seq2seq	20.0%	22.0%	6.5%	1.5%
HLSTM	21.0%	17.0%	10.0%	3.5%
HLSTM+Copy	28.0%	19.0%	7.0%	5.5%
HLSTM+MSA(GA)	33.5%	25.0%	5.0%	4.0%
HLSTM+MSA(CA)	30.0%	26.0%	2.0%	8.0%
IE (ours)	36.0%	**34.0%**	2.0%	4.0%
IE+MSA(GA) (ours)	**45.0%**	24.0%	5.0%	2.0%
IE+MSA(CA) (ours)	41.0%	27.0%	4.0%	2.0%

Table 2: Data distribution over Gram.-Logic. scores. a-b denotes that the grammar score is a and the logicality score is b. Each cell denotes the proportion of the endings with score a-b.

Examples and Attention Visualization

We presented an example of generated story endings in Table 3. Our model generates more natural and reasonable endings than the baselines.

In this example, the baselines predicted wrong events in the ending. Baselines (Seq2Seq, HLSTM,

Context:	Martha is **cooking** a special **meal** for her family. She **wants everything to be just right** for when they eat. Martha **perfects everything** and puts her **dinner** into the **oven**. Martha goes to **lay down** for a quick **nap**.
Golden Ending:	She **oversleeps** and runs into the **kitchen** to take out her **burnt dinner**.
Seq2Seq:	She was so happy to have a ***new cake***.
HLSTM:	Her family ***and her family*** are very happy with her **food**.
HLSTM+ Copy:	**Martha** is happy to be able to ***eat her family***.
HLSTM+ GA:	She is happy to be able to **cook her dinner**.
HLSTM+ CA:	She is very happy that she has made a new **cook** .
IE:	She is very happy with her **family**.
IE+GA:	When she gets back to the **kitchen**, she sees a **burning light** on the **stove**.
IE+CA:	She realizes the **food** and is happy she was ready to **cook** .

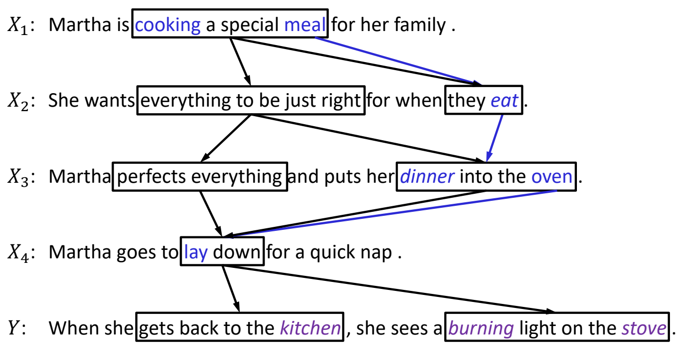
Table 3: Generated endings from different models. **Bold** words denote the **key** entity and event in the story. *Improper* words in ending is in *italic* and proper words are underlined.

and HLSTM+Copy) have predicted improper entities (*cake*), generated repetitive contents (*her family*), or copied wrong words (*eat*). The models equipped with incremental encoding or knowledge through MSA(GA/CA) perform better in this example. The ending by IE+MSA is more coherent in logic, and fluent in grammar. We can see that there may exist multiple reasonable endings for the same story context.

In order to verify the ability of our model to utilize the context clues and implicit knowledge when planning the story plot, we visualized the attention weights of this example, as shown in Figure 3. Note that this example is produced from graph attention.

In Figure 3, phrases in the box are key events of the sentences that are manually highlighted. Words in blue or purple are entities that can be retrieved from ConceptNet, respectively in story context or in ending. An arrow indicates that the words in the current box (e.g., *they eat* in X_2) all have largest attention weights to some words in the box of the preceding sentence (e.g., *cooking a special meal* in X_1). Black arrows are for state context vector (see Eq.25) and blue for knowledge context vector (see Eq.26). For instance, *eat* has the largest knowledge attention to *meal* through the knowledge graph (<*meal, AtLocation, dinner*>,<*meal, RelatedTo, eat*>). Similarly, *eat* also has the second largest attention weight to *cooking* through the knowledge graph. For attention weights of state context vector, both words in *perfects everything* has the largest weight to some of *everything to be just right* (*everything*→*everything*, *perfect* → *right*).

The example illustrates how the connection between con-


X_1: Martha is cooking a special meal for her family .
X_2: She wants everything to be just right for when they eat .
X_3: Martha perfects everything and puts her dinner into the oven .
X_4: Martha goes to lay down for a quick nap .
Y: When she gets back to the kitchen , she sees a burning light on the stove .

Entity	commonsense knowledge
cook	(cook, AtLocation, *kitchen*)
	(cook, HasLastSubevent, *eat*)
meal	(meal, AtLocation, *dinner*)
	(meal, RelatedTo, *eat*)
eat	(eat, AtLocation, *dinner*)
oven	(oven, AtLocation, *stove*)
	(oven, RelatedTo, *kitchen*)
	(oven, UsedFor, *burn*)

Figure 3: An example illustrating how incremental encoding builds connections between context clues.

text clues are built through incremental encoding and use of commonsense knowledge. The chain of context clues, such as $be\_cooking \rightarrow want\_everything\_be\_right \rightarrow perfect\_everything \rightarrow lay\_down \rightarrow get\_back$, and the commonsense knowledge, such as <*cook, AtLocation, kitchen*> and <*oven, UsedFor, burn*>, are useful for generating reasonable story endings.

Conclusion and Future Work

We present a story ending generation model that builds context clues via incremental encoding and leverages commonsense knowledge with multi-source attention. It encodes a story context incrementally with a multi-source attention mechanism to utilize not only context clues but also commonsense knowledge: when encoding a sentence, the model obtains a multi-source context vector which is an attentive read of the words and the corresponding knowledge graphs of the preceding sentence in the story context. Experiments show that our models can generate more coherent and reasonable story endings.

As future work, our incremental encoding and multi-source attention for using commonsense knowledge may be applicable to other language generation tasks.

Refer to the Appendix for more details.

Acknowledgements

This work was jointly supported by the National Science Foundation of China (Grant No.61876096/61332007), and the National Key R&D Program of China (Grant No. 2018YFC0830200). We would like to thank Prof. Xiaoyan Zhu for her generous support.

References

[Bahdanau, Cho, and Bengio 2014] Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Cai, Tu, and Gimpel 2017] Cai, Z.; Tu, L.; and Gimpel, K. 2017. Pay attention to the ending:strong neural baselines for the roc story cloze task. In *ACL*, 616–622.

[Chaturvedi, Peng, and Dan 2017] Chaturvedi, S.; Peng, H.; and Dan, R. 2017. Story comprehension for predicting what happens next. In *EMNLP*, 1603–1614.

[Clark, Ji, and Smith 2018] Clark, E.; Ji, Y.; and Smith, N. A. 2018. Neural text generation in stories using entity representations as context. In *NAACL*, 1631–1640.

[Dagan, Glickman, and Magnini 2006] Dagan, I.; Glickman, O.; and Magnini, B. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, 177–190.

[Fan, Lewis, and Dauphin 2018] Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical neural story generation. In *ACL*, 889–898.

[Gu et al. 2016] Gu, J.; Lu, Z.; Li, H.; and Li, V. O. K. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, 1631–1640.

[Hermann et al. 2015] Hermann, K. M.; Kočiský, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *NIPS'15*, 1693–1701.

[Jain et al. 2017] Jain, P.; Agrawal, P.; Mishra, A.; Sukhwani, M.; Laha, A.; and Sankaranarayanan, K. 2017. Story generation from sequence of independent short descriptions. *arXiv preprint arXiv:1707.05501*.

[Ji et al. 2017] Ji, Y.; Tan, C.; Martschat, S.; Choi, Y.; and Smith, N. A. 2017. Dynamic entity representations in neural language models. In *EMNLP*, 1830–1839.

[Knight et al. 2018] Knight, K.; Choi, Y.; Sap, M.; Rashkin, H.; and Bosselut, A. 2018. Modeling naive psychology of characters in simple commonsense stories. In *ACL*, 2289–2299.

[Li et al. 2013] Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. O. 2013. Story generation with crowdsourced plot graphs. In *AAAI*, 598–604.

[Li et al. 2018] Li, Q.; Li, Z.; Wei, J.-M.; Gu, Y.; Jatowt, A.; and Yang, Z. 2018. A multi-attention based neural network with external knowledge for story ending predicting task. In *COLING*, 1754–1762.

[Li, Ding, and Liu 2018] Li, Z.; Ding, X.; and Liu, T. 2018. Generating reasonable and diversified story ending using sequence to sequence model with adversarial training. In *COLING*, 1033–1043.

[Lin, Sun, and Han 2017] Lin, H.; Sun, L.; and Han, X. 2017. Reasoning with heterogeneous knowledge for commonsense machine comprehension. In *EMNLP*, 2032–2043.

[LoBue and Yates 2011] LoBue, P., and Yates, A. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *ACL*, HLT '11, 329–334.

[Luong, Pham, and Manning 2015] Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, 1412–1421.

[Martin et al. 2018] Martin, L.; Ammanabrolu, P.; Wang, X.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. 2018. Event representations for automated story generation with deep neural nets. In *AAAI*, 868–875.

[Mihaylov and Frank 2018] Mihaylov, T., and Frank, A. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *ACL*, 821–832.

[Mostafazadeh et al. 2016a] Mostafazadeh, N.; Chambers, N.; He, X.; Parikh, D.; Batra, D.; Vanderwende, L.; Kohli, P.; and Allen, J. 2016a. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL*, 839–849.

[Mostafazadeh et al. 2016b] Mostafazadeh, N.; Vanderwende, L.; Yih, W. T.; Kohli, P.; and Allen, J. 2016b. Story cloze evaluator: Vector space representation evaluation by predicting what happens next. In *The Workshop on Evaluating Vector-Space Representations for Nlp*, 24–29.

[Papineni et al. 2002] Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W. J. 2002. Ibm research report bleu: a method for automatic evaluation of machine translation. In *ACL*, 311–318.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

[R. Bowman et al. 2015] R. Bowman, S.; Angeli, G.; Potts, C.; and Manning, C. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, 632–642.

[Rashkin et al. 2018] Rashkin, H.; Sap, M.; Allaway, E.; Smith, N. A.; and Choi, Y. 2018. Event2mind: Commonsense inference on events, intents, and reactions. In *ACL*, 463–473.

[Roemmele 2016] Roemmele, M. 2016. Writing Stories with Help from Recurrent Neural Networks. In *AAAI*, 4311 – 4312. Phoenix, AZ: AAAI Press.

[Soo, Lee, and Chen 2016] Soo, V.; Lee, C.; and Chen, T. 2016. Generate believable causal plots with user preferences using constrained monte carlo tree search. In *AAAI*, 218–224.

[Speer and Havasi 2012] Speer, R., and Havasi, C. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, 3679–3686.

[Sutskever, Vinyals, and Le 2014] Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3104–3112.

[Wang, Liu, and Zhao 2017] Wang, B.; Liu, K.; and Zhao, J. 2017. Conditional generative adversarial networks for commonsense machine comprehension. In *IJCAI*, 4123–4129.

[Yang et al. 2016] Yang, Z.; Yang, D.; Dyer, C.; He, X.;

Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *NAACL*, 1480–1489.

[Zhang et al. 2017] Zhang, S.; Rudinger, R.; Duh, K.; and Van Durme, B. 2017. Ordinal common-sense inference. In *ACL*, 379395.

[Zhao et al. 2018] Zhao, Y.; Liu, L.; Liu, C.; Yang, R.; and Yu, D. 2018. From plots to endings: A reinforced pointer generator for story ending generation. In Zhang, M.; Ng, V.; Zhao, D.; Li, S.; and Zan, H., eds., *NLPCC*, 51–63.

[Zhou et al. 2018] Zhou, H.; Yang, T.; Huang, M.; Zhao, H.; Xu, J.; and Zhu, X. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*.

Appendix A: Annotation Statistics

We presented the statistics of annotation agreement in Table 4. The proportion of the annotations in which at least two annotators (3/3+2/3) assigned the same score to an ending is 96% for grammar and 94% for logicality. We can also see that the 3/3 agreement for logicality is much lower than that for grammar, indicating that logicality is more complicated for annotation than grammar.

Metric	3/3	2/3	1/3
Gram.	58%	38%	4%
Logic.	38%	57%	6%

Table 4: Annotation statistics. $n/3$ agreement means n out of 3 annotators assigned the same score to an ending.

Appendix B: Error Analysis

We analyzed error types by manually checking all 46 bad endings generated by our model, where *bad* means the average score in terms of at least one metric is not greater than 1.

There are 3 typical error types: *bad grammar* (BG), *bad logicality* (BL), and other errors. The distribution of types is shown in Table 5.

Error Type	BG	BL	Others
IE+MGA	32.6%	37.0%	30.4%

Table 5: Distribution of errors.

We also presented some typical cases for each error type in Table 6. Note that we only took *graph attention* as example. The first case shows an instance of bad grammar for repetitive generation. The second case shows that our model predicted a wrong entity at the last position where *car* is obviously more appropriate than *daughter*. It happens when the attention focuses on the wrong position, but in more cases it happens due to the noise of the commonsense knowledge base[5]. The ending of the third case contains a relevant event *work on his own* but the event is not consistent to the previous word *relieved*. Other cases show that our model is not good at dealing with rare words. However, this can be further improved by applying copy mechanism, as our future work. These errors also indicate that story ending generation is challenging, and logic and implicit knowledge plays a central role in this task.

[5]Note that ConceptNet contains many low-quality triples.

Context:	Sean **practiced driving** every day. He **studied hard** for the written part of the **driving test**. Sean felt ready to **take his driving test** after a couple months. Sean was confident that he would **pass his driving test**.
Golden Ending:	Sean <u>**passed his driving test**</u> on the first attempt .
IE+MSA (GA):	He <u>**failed his test**</u> ***and failed his test***.
Context:	Danny wanted to give his **daughter** a sweet **gift**. DJ was **turning 18 years old** and ready to **drive**. Danny decided it was time for **a new car**. Danny went down to the dealership and **bought a new car**.
Golden Ending:	He <u>**surprised**</u> DJ on her <u>**18th birthday**</u> with a <u>**new car**</u>.
IE+MSA (GA):	He <u>**was so happy with**</u> his ***new daughter***.
Context:	There was a big **pothole** near Bill's house. He had to **swerve** to avoid it all the time. He kept **calling the city** to **fix** it. Eventually someone **returned his calls**.
Golden Ending:	It took three more months for it to <u>**be fixed**</u>
IE+MSA (GA):	He was ***relieved*** to find that he had to <u>**work on his own**</u>.

Table 6: Typical cases of errors. Bold words denote the key entity and event in the story. Improper words in ending is in italic and proper is underlined.

Appendix C: Attention Visualization

The multi-source attention mechanism computes the state context vectors and knowledge context vectors respectively as follows:

$$\mathbf{c}_{\mathbf{h}j}^{(i)} = \sum_{k=1}^{l_{i-1}} \alpha_{h_k,j}^{(i)} \mathbf{h}_k^{(i-1)}, \tag{25}$$

$$\mathbf{c}_{\mathbf{x}j}^{(i)} = \sum_{k=1}^{l_{i-1}} \alpha_{x_k,j}^{(i)} \mathbf{g}(x_k^{(i-1)}), \tag{26}$$

The visualization analysis in Section 4.6 "Generated Ending Examples and Attention Visualization" is based on the attention weights ($\alpha_{h_{k,j}}^{(i)}$ and $\alpha_{x_{k,j}}^{(i)}$), as presented in Figure 4. Similarly we take as example the graph attention method to represent commonsense knowledge.

The figure illustrates how the incremental encoding scheme with the multi-source attention utilizes context clues

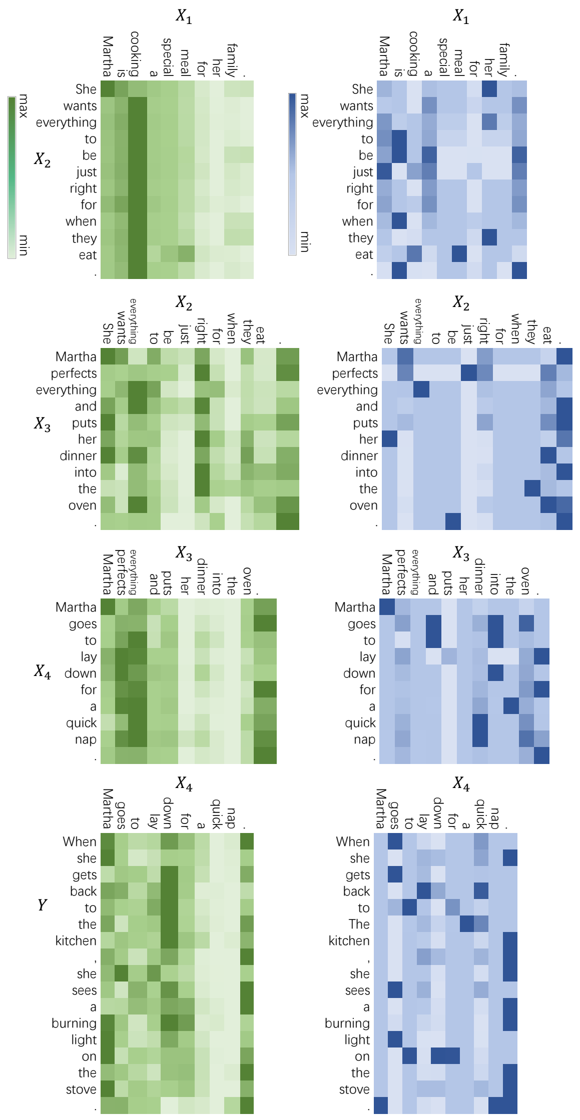


Figure 4: Attention weight visualization by multi-source attention. Green graphs are for state context vectors (Eq. 25) and blue for knowledge context vectors (Eq. 26). Each graph represents the attention weight matrix for two adjacent sentences (X_i in row and X_{i+1} in column). The five graphs in the left column show the state attention weights in the incremental encoding process.

and implicit knowledge.

1) The left column: for **utilizing context clues**, when the model encodes X_2, *cooking* in X_1 obtains the largest state attention weight ($\alpha^{(i)}_{h_{k,j}}$), which illustrates *cooking* is an important word (or event) for the context clue. Similarly, the key events in each sentence have largest attention weights to some entities or events in the preceding sentence, which forms the context clue (e.g., *perfects* in X_3 to *right* in X_2, *lay/down* in X_4 to *perfect/everything* in X_3, *get/back* in Y to *lay/down* in X_4, etc.).

2) The right column: for the use of **commonsense knowledge**, each sentence has attention weights ($\alpha^{(i)}_{x_{k,j}}$) to the knowledge graphs of the preceding sentence (e.g. *eat* in X_2 to *meal* in X_1, *dinner* in X_3 to *eat* in X_2, etc.). In this manner, the knowledge information is added into the encoding process of each sentence, which helps story comprehension for better ending generation (e.g., *kitchen* in Y to *oven* in X_2, etc.).