# An Ion Exchange Mechanism Inspired Story Ending Generator for Different Characters

Xinyu Jiang[1⋆], Qi Zhang[2,3⋆], Chongyang Shi✉[1], Kaiying Jiang[4], Liang Hu[3,5], and Shoujin Wang[6]

[1] Beijing Institute of Technology
[2] University of Technology Sydney
[3] Deepblue Academy of Sciences
[4] University of Science and Technology Beijing
[5] Tongji University
[6] Macquarie University
jiangxy_08@163.com,{zhangqi_cs,cy_shi}@bit.edu.cn

**Abstract.** Story ending generation aims at generating reasonable endings for a given story context. Most existing studies in this area focus on generating coherent or diversified story endings, while they ignore that different characters may lead to different endings for a given story. In this paper, we propose a Character-oriented Story Ending Generator (CoSEG) to customize an ending for each character in a story. Specifically, we first propose a character modeling module to learn the personalities of characters from their descriptive experiences extracted from the story context. Then, inspired by the ion exchange mechanism in chemical reactions, we design a novel vector breaking/forming module to learn the intrinsic interactions between each character and the corresponding context through an analogical information exchange procedure. Finally, we leverage the attention mechanism to learn effective character-specific interactions and feed each interaction into a decoder to generate character-orient endings. Extensive experimental results and case studies demonstrate that CoSEG achieves significant improvements in the quality of generated endings compared with state-of-the-art methods, and it effectively customizes the endings for different characters.

**Keywords:** Story Ending Generation · Character-Oriented · Neural Network

## 1 Introduction

Story ending generation aims to deliver a comprehensive understanding of the context and predict the next plot for a given story [10,16,31,29]. Some studies in this field generate coherent stories by modeling the sequence of events or verbs [5,19], or diversify story generation by introducing common senses or vocabulary

---

⋆ The first two authors contribute equally to this work.
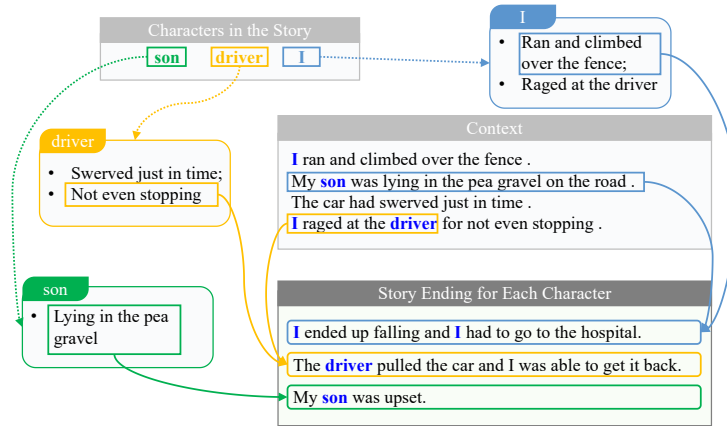✉ Corresponding author

Fig. 1: An example of the story context in the ROCStories corpus, and the endings generated by our model for different characters.

information [8,18]. Others focus on controlling the sentiment of story endings [16,27] or generating the missing plot for an incomplete story [1,29]. These methods generally ignore the relation and interaction between story plots and characters and simplify the influence of character personality on story generation, leading to desirable but character-irrelevant story endings.

Intuitively, stories are derived from characters, and character personality directly determines the plot and direction of the story. Figure 1 shows an example of a typical story in the ROCStories corpus [21] and the endings generated for different characters. From the figure, we can observe that: **1)** each character has its unique personality depicted by its character token and character experience, i.e., the character-related descriptions in a story. For example, the character **son** is depicted by the token "son" and the description "lying in the pea gravel"; **2)** naturally, different characters with different personalities interact with the story context and thus affect the story plot, leading to different story endings (see the different endings for 'son', 'driver' and 'I' in the example).

Customizing the endings for different characters in a story is a novel but challenging task since there is no one-to-many dataset (i.e., one story corresponds to many ground-truth endings). To the best of our knowledge, most previous methods for the story ending generation aim to generate a single ending or missing plot rather than diverse coherent endings of different characters, for a given story context [10,29,31]. The main challenges in customizing character-oriented story endings are 1) *to model the personality of each character*, and 2) *to learn the diverse interactions between different characters and the story context*. Intuitively, a story context contains a character's experiences, i.e., the multiple descriptions of the character, which depict the personalities of the character. It would be helpful to extract the related descriptions of each character from the story content
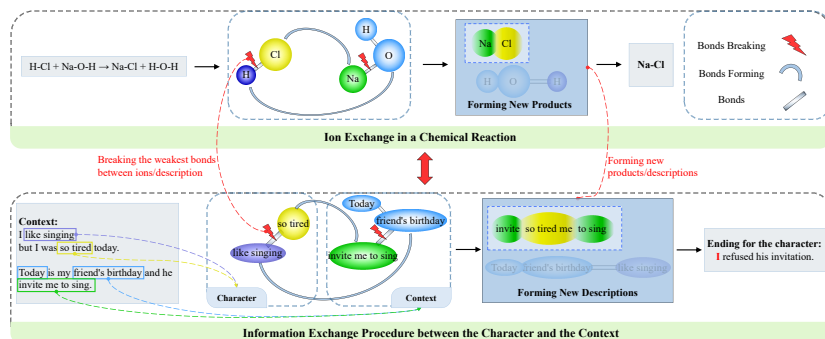
Fig. 2: A character-context information exchange mechanism to learn the interaction between character and context, which is inspired by the ion exchange mechanism in chemical reactions.

and build its experience sequence via organizing the descriptions in chronological order for modeling each character's personalities.

Inspired by recent studies using deep learning to plan and predict chemical reactions [26,25], we model the personalities of characters by analogizing the interactions between characters and context to chemical reactions. Specifically, we believe that the information exchange between different characters and context in generating new situational (character-specific) semantics during the interaction is similar to the ion exchange [2] to form new products in chemical reactions (cf. Figure 2). Derived from this observation, it is promising to learn the interaction between a character and the corresponding context following an information exchange procedure. As depicted in Figure 2, by exchanging related descriptions of the character and context, a new informative and character-related description "invite so tired me to sing" is formed by putting "so tired" (from character) and "invite me to sing" (from context) together. Consequently, the newly formed character-related description leads to an ending "refused his invitation" customized for the character "I" with a high probability.

Accordingly, we propose a Character-oriented Story Ending Generator (CoSEG) to customize an ending for each character in a story. Specifically, the proposed model first learns a representation of each character's personality by modeling its experiences with a Character Modeling module (CMM) and a context representation by modeling the story content. Then, a novel Vector Breaking/Forming module (VBF) is introduced to effectively learn the interaction between each character and the context through multiple information exchanges. Finally, a character-specific interaction representation is generated by adaptively picking out the most effective interaction via a Character-Context Attention module (C-CA), and each interaction representation is further utilized to customize the ending for the corresponding character. Note that CoSEG adopts LSTM-based encoder-decoder architecture, and the proposed key modules are network-agnostics and are

also suitable for other prevailing networks, e.g., CNN [14] and Transformer [32]. The main contributions of our paper are summarized below:

- We propose a Character-oriented Story Ending Generator (CoSEG) model to tackle the challenging task of customizing story endings for different characters.
- We introduce a character modeling module to effectively model the personality of each character and learn a personalized and informative character representation.
- Inspired by the ion exchange process in chemical reactions, we propose a novel VBF module to learn the interaction between the character and context based on the information exchange mechanism.

Extensive experimental results on the ROCStories dataset show that our proposed CoSEG not only generates more coherent and diversified story endings compared with state-of-the-art and/or representative baseline methods, but also customizes effective endings for each character in a story. The superiority of CoSEG also demonstrates the effectiveness of the proposed CMM and VBF module in customizing story endings.

## 2   Related Work

Neural network-based models are the current mainstream in story generation methods owing to their impressive generation performance [22,6,31,30,18]. In recent years, there have been many innovations that utilize the encoder and decoder framework to generate coherent and diversified story endings. [33] applies a hierarchical attention architecture to encode text information to generate the context representation. [19] predicts the next event by extracting the event represented from the sentence, thereby ensuring the coherence of the story. [5] uses one head of the decoder's self-attention to attend only to previously generated verbs in order to generate a coherent story. [4] learns a second seq2seq model, which is led by the first model to focus on what the first model failed to learn. [10] introduces external knowledge and utilizes an incremental encoding scheme to ensure the diversity of stories. In addition, recent work proposes a character-centric neural storytelling model to generate stories for a given character [15]. Excited with the excellent performance of attention-based models [34,7] like Transformer [28] and BERT [3] in recent years, many story-ending generation and completion models leverage self-attention mechanism and Transformer architecture to enhance the quality of generated story endings [9,32]. In this work, we adopt LSTM as the backbone in the model design and experiments. Other architectures, e.g., attention networks and Transformer, can easily be incorporated into the proposed CoSEG.

However, most of the aforementioned generation methods cannot generate multiple coherent and diverse endings for a single story context. Moreover, only a few works focus on generating multiple endings or responses given a single context. [8] uses several unobserved latent variables $z$ to generate different responses. This

method, however, relies on a one-to-many dataset. [16] applies an additional sentiment analyzer to first predict the sentiment intensity $s$ of the ground truth ending $y$, then constructs paired data $(x, s; y)$ for training, where $x$ is the story context. In the generation stage, the model receives the sentiment variable $s$ from users to generate a sentiment-specific ending. Recent work [29,20] introduce prevalent Transformers to learn story representation for generating a missing plot or a story ending. All these methods assume that the plot has little relation or interaction with the personality of the characters in the story. Unlike these models, our proposed model can customize an ending for each character in the story context without relying on a one-to-many dataset.

## 3   Character-oriented Story Ending Generator (CoSEG)

### 3.1   Problem Definition and Architecture

In this section, we formulate the task of customizing the ending for each character in a story. Given a story content $x = (x_1, ..., x_l)$, which contains $l$ sentences, and $m$ characters $(c_1, ..., c_m)$. The task is to predict customized endings $y = (y_1, ..., y_m)$ for all the $m$ characters.

A story generally corresponds to only one ground truth ending, which may consist of the actions or opinions of a particular character; the endings of other characters are unavailable. To tackle the issue, in the training stage, we extract the experience sequence of the character in the ground-truth ending, then train our proposed model to generate an ending related to the extracted character (ground-truth ending). In the generating stage, we extract the experience sequences of all characters who appear in the story and then apply the proposed model to generate an ending for the characters.

The architecture of our proposed CoSEG model is depicted in Figure 3. Our model consists of three modules—a CMM module, a VBF module and a C-CA module—as well as an encoder to encode the story context and a decoder to generate the story ending. As shown in Figure 3, the CMM module generates the character representation $cc$ for each character $c_i$ by modeling the character's experiences; the VBF module learns the interaction between the character representation and the story context through multiple information exchanges and generates multiple interaction results, namely product candidates $(\mathbf{p}_0^c, ..., \mathbf{p}_n^c)$; the C-CA module generates a character-related story context representation by picking out the most effective product candidates. The context representation is further used as the initial state of the decoder to predict an ending for the character. The following sections present the details of each module.

### 3.2   Character Experience Sequences

We construct the experience sequence for each character in the story; the experience sequence is further fed into CMM to generate the character representation.
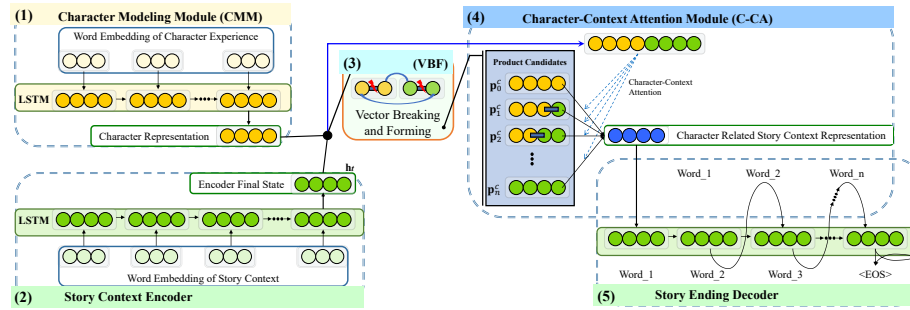
Fig. 3: Character-oriented Story Ending Generator (CoSEG).

We take the sentence *She knew a discount store near her sold socks* as an example to illustrate how to extract a character experience with the following four steps:

1) Construct a **dependency tree** for the given sentence, and get the headword *knew* of the character *She*.

2) Extract the **context words**, namely *knew, discount, store, sold, socks*, as the first part of the character experience. This part is the background of the story.

3) Extract the **entity words**, namely the character *She*, the headword *knew* and the corresponding object *store*, as the second part of the character experience.

4) Connect the above two parts of character experience with a token *OBJ* to obtain the final character experience [ *knew, discount, store, sold, socks, OBJ, knew, store, She* ]. The token *OBJ* is utilized to separate the two parts, explicitly telling the model which words are context information and which are directly related to the character.

In this way, we can extract each character's experience from each sentence. Subsequently, we build the experience sequence $(e_1^i, ..., e_s^i)$ of the character $c_i$ via organizing the character's experience in chronological order, where $s$ is the number of the sentences that contain $c_i$.

### 3.3   Character Modeling Module (CMM)

We generate the character representation by modeling the character experience sequence. Formally, given a character $c_i$ and a corresponding character experience sequence $(e_1^i, ..., e_S^i)$, the module encodes the token sequence $(w_1^s, ..., w_{T_s}^s)$ inside each experience $e_s^i$ to obtain the hidden states $(\mathbf{h}_1^s, ..., \mathbf{h}_{T_s}^s)$, where $T_s$ is the length of $e_s^i$, and the superscript $s$ of $\mathbf{h}$ or $w$ represents it is for the $s^{th}$ character experience.

We choose the final hidden state $\mathbf{h}_{T_s}^s$ as the representation of the character $c_i$ who went through the character experience $e_s^i$. The character representation $\mathbf{h}_{T_s}^s$ is then used as the initial state of the next encoder to generate further enriched

character representation $\mathbf{h}_{T_{s+1}}^{s+1}$ as follows:

$$\mathbf{h}_t^{s+1} = \mathbf{LSTM}(\mathbf{h}_{t-1}^{s+1}, \mathbf{w}_t^{s+1}, \mathbf{h}_{T_s}^s), \tag{1}$$

where $T_{s+1}$ is the length of experience $e_{s+1}^i$, $w_t^{s+1}$ is the $t^{th}$ token in the sequence $(w_1^{s+1}, ..., w_{T_{s+1}}^{s+1})$ inside the experience $e_{s+1}^i$, $\mathbf{w}_t^{s+1}$ denotes the embedding of $w_t^{s+1}$ and the superscript $s+1$ of $\mathbf{h}$ or $w$ represents it is for the $(s+1)^{th}$ character experience.

Finally, the CMM Module will generate the $S^{th}$ character representation $\mathbf{h}_{T_S}$, which has gone through all the character experiences $(e_1^i, ..., e_S^i)$.

### 3.4   Vector Breaking and Forming (VBF)

Since both character and context are represented with high-dimensional vectors, we propose a novel VBF module to learn the interaction between the character and the story context representation based on the information exchange procedure (cf. Figure 2) and then generate multiple product candidates.

**Vector Breaking:** Assume that there are invisible bonds between the adjacent elements of a vector, and VBF breaks the bonds between adjacent elements. For a vector of size $n$, there are $n+1$ potential bond-breaking positions. For example, given a vector $v_1 = [0.1, 0.2]$, the size of $v_1$ is 2 and we have 3 bond-breaking positions, as follows:

$$
\begin{aligned}
v_1^l, v_1^r &= [], [0.1, 0.2] = VecB_0(v_1), \\
v_1^{l'}, v_1^{r'} &= [0.1], [0.2] = VecB_1(v_1), \\
v_1^{l''}, v_1^{r''} &= [0.1, 0.2], [] = VecB_2(v_1),
\end{aligned}
\tag{2}
$$

where $VecB_k$ represents breaking the bond in the position $k$, and the superscripts $l$ and $r$ of the breaking results represent the *left* and *right* parts of $v_1$, respectively.

**Vector Forming:** The two interaction vectors break at each position respectively. To keep the size of interaction results constant, for each bond-breaking position, VBF integrates the left part of the first vector and the right part of the second vector to generate a product candidate[7]. In this way, two vectors of size $n$ can interact to obtain a total of $n+1$ product candidates. For example, let $v_1$ interact with $v_2 = [0.3, 0.4]$, and we can obtain such three product candidates $\mathbf{p}_0^c = [0.3, 0.4], \mathbf{p}_1^c = [0.1, 0.4], \mathbf{p}_2^c = [0.1, 0.2]$. As shown in Figure 3 part (3), the character representation and the encoder final state $\mathbf{h}_t$ (i.e., story context representation) interact in the VBF module to generate the product candidates $(\mathbf{p}_0^c, ..., \mathbf{p}_n^c)$.

### 3.5   Character-Context Attention (C-CA)

As shown in Figure 3 part (4), the C-CA Module aims to pick out the most effective product candidates. Specifically, we utilize the $s^{th}$ character representation $\mathbf{h}_{T_s}$

---

[7] There is no order between the two interaction vectors, which vector as the first one has little influence on the experimental results.

and the encoder final state $\mathbf{h}_t$ to obtain the attention weight of each product candidate $\mathbf{p}_k^c$:

$$\mathbf{a}^s = \sigma(\mathbf{W}_a[\mathbf{h}_{T_s}; \mathbf{h}_t] + \mathbf{b}_a),$$
$$\mathbf{r}^s = \sum_{k=0}^{n} \mathbf{a}_k^s \mathbf{p}_k^c, \tag{3}$$

where $\sigma$ is the softmax function, $\mathbf{W}_a$ is the weight matrix, $\mathbf{b}_a$ is the bias, $att$ is the attention weight, and $\mathbf{a}^s$ stands for the character-related story context representation of the $s^{th}$ character. As shown in Figure 3 part (5), the $\mathbf{r}^s$ is further used as the initial state of the decoder (note that we omit the superscript $s$ in the following for simplicity):

$$\mathbf{h}_t^y = \mathbf{LSTM}(\mathbf{h}_{t-1}^y, \mathbf{w}_{t-1}^y, \mathbf{r}), \tag{4}$$

where $\mathbf{h}_t^y$ is the $t^{th}$ hidden state of the decoder, which is further utilized to generate the $t^{th}$ word $w_t^y$, $\mathbf{w}_{t-1}^y$ is the embedding of word $w_{t-1}^y$.

## 4   Experiment

In this section, we conduct extensive experiments to investigate the quality of CoSEG and the comparative baselines.

### 4.1   Dataset

We evaluated our model on the ROCStories corpus [21]. This corpus contains 98,162 five-sentence stories. Our task is to generate an ending for each character that appears in a given four-sentence story[8]. For each story, we extract the experience sequence for the character who appears in the ground truth ending. We select 66,881 stories in which the length of the ground-truth character's experience sequence is no less than 2 and treat these stories as the training set. We elaborately design two test sets, each with 3073 stories. Specifically, the two sets are called sufficient test set and inadequate test set. In the sufficient test set, the length of the ground-truth character's experience sequence is no less than 2 for all stories, while the length is less than 2 for all stories in the inadequate test set. The two test sets are applied to evaluate the performance of our proposed model when the character information is sufficient and inadequate, respectively.

---

[8] We identify characters in a macro way. We extract the subject of each sentence in the story. We regard *name entity* or *noun* as the character of the sentence. In principle, in this way, we can generate an ending for any *noun*. Since there are few endings regarding nouns as the characters in the training data (such as the ending with "car" as the character), the proposed model is difficult to generate high-quality endings for those characters.

## 4.2   Experimental Settings

We use the GloVe.6B [24] pre-trained word embedding, and the number of dimensions is 200. The hidden size of the LSTM cell is 512. Since the size of the character representation and the encoder's final state both are 512, the number of product candidates will be $512 + 1 = 513$. A larger dimension size brings large computation costs for the model and the device. In summary, the detailed experimental settings are provided as follows:

- We use 66,881 stories for training.
- We use 3073 stories for validation, which is shared by the two test sets.
- We have two test sets with 3073 stories each. We refer to the two test sets as the sufficient test set and the inadequate test set respectively. In the sufficient test set, the character in the ground truth ending also appears multiple times in the story context, and in the inadequate test set is the opposite. These two test sets evaluate the performance of our proposed model when the character information is sufficient and inadequate, respectively.
- We use Momentum Optimizer to update parameters when training and empirically set the momentum to be 0.9.
- The size of the character representation and the encoder hidden state are 512.
- We select the product candidates generated using $VecB_0$, $VecB_{128}$, $VecB_{256}$, $VecB_{384}$ and $VecB_{512}$ five breaking operation.
- The number of product candidates will be $512 + 1 = 513$.

Note that the reason for the selection of the product candidates is analyzed in *Combination Analysis on Product Candidates* in Section 4.5.

## 4.3   Baselines

We compared our model with the following state-of-the-art baseline methods:

**Seq2Seq** [17]: A vanilla encoder-decoder model with an attention mechanism. The model treats the story context as a single sentence.

**HAN** [33]: A hierarchical attention architecture is applied to encode text information so as to generate the context representation.

**IE** [10]: It adopts an incremental encoding scheme to represent context clues and applies commonsense knowledge by multi-source attention.

**T-CVAE** [29]: It proposes a conditional variational autoencoder based on Transformer for missing plot generation.

**MGCN-DP** [11]: It leverages multi-level graph convolutional networks over dependency parse trees to capture dependency relations and context clues.

In addition, we introduce two variations of the proposed CoSEG model:

**CoADD**: We replace the VBF Module in CoSEG with an element-wise summation.

**CoCAT**: We concatenate the character representation and the encoder final state, and pass the concatenated vector through a linear layer to obtain the character-related story context representation.

### 4.4   Evaluation Metrics

We evaluate our model from two perspectives: the quality of the generated endings and the ability to customize endings.

**Quality Evaluation**  We adopt two kinds of evaluations to investigate the ability of the proposed method and the baselines in generating high-quality story endings.

**Automatic Evaluation:** We use perplexity (PPL) and BLEU (BLEU-1, BLEU-2 and BLEU-3) [23] to evaluate the quality of the generated endings. A smaller PPL and a higher BLEU indicate a better ending.

**Manual Evaluation:** We hire three evaluators, who are experts in English, to evaluate the generated story endings. We randomly sampled 200 stories from the two test sets and obtained 1400 endings from the seven models for each test set. Evaluators need to score the generated endings in terms of two criteria: coherency and grammar. The coherency score measures whether the endings are coherent with the story context; specifically, the score of 3 denotes coherency, the score of 1 denotes coherency to some extent, and the score of 0 denotes no coherency at all. In addition, the grammar score measures whether there are grammatical errors in generated endings; a grammar score is 0 if endings have errors, and 1 otherwise.

**Ability to Customize Endings**  We randomly sample 200 stories from the two test sets and generate ending for one random character in each story. To evaluate the ability to customize endings for different characters, we propose three evaluation metrics:

**Success Rate (SucR):** SucR measures whether the subject of the generated ending is the selected character.

**Rationality:** We adopt three levels to evaluate whether the generated ending matches the selected character given the story context: level 3 denotes perfect matching, level 1 denotes partial matching, and level 0 for mismatching.

**Discrimination Degree (DiscD):** Given an ending generated by our proposed model, we further hire three evaluators to choose which character is the ending generated for. If the character chosen by the evaluator is consistent with the selected character, it scores 1; and 0 otherwise.

### 4.5   Evaluation Results

*Automatic Evaluation*  The automatic evaluation results for the sufficient and inadequate test sets are shown in Table 1. From the table, we can observe the following:

1) In both the sufficient and inadequate test sets, our model has lower perplexity and higher BLEU scores than the baselines. Specifically, in terms of perplexity, **CoSEG** outperforms **MGCN-DP**, **T-CVAE**, **IE**, **HAN** and **Seq2Seq** by 1.02/ 1.22/ 2.09/ 3.44/ 3.27 respectively in the sufficient test set,

Table 1: Automatic evaluation results of the sufficient and the inadequate test set.

| Sufficient | | | | |
|---|---|---|---|---|
| Model | PPL | BLEU-1 (%) | BLEU-2 (%) | BLEU-3 (%) |
| Seq2Seq | 13.26 | 22.46 | 6.88 | 4.21 |
| HAN | 13.43 | 22.43 | 6.96 | 4.47 |
| IE | 12.08 | 23.08 | 7.43 | 4.67 |
| T-CVAE | 11.21 | 23.72 | 8.05 | 5.11 |
| MGCN-DP | 11.01 | 23.90 | 8.11 | 5.34 |
| CoADD | 12.14 | 23.92 | 7.74 | 4.68 |
| CoCAT | 11.45 | 24.26 | 8.53 | 5.41 |
| CoSEG | **9.99** | **25.28** | **9.10** | **5.93** |
| Inadequate | | | | |
| Model | PPL | BLEU-1 (%) | BLEU-2 (%) | BLEU-3 (%) |
| Seq2Seq | 21.81 | 17.13 | 3.76 | 1.76 |
| HAN | 24.26 | 17.15 | 4.08 | 2.32 |
| IE | 16.90 | 18.40 | 4.89 | 2.78 |
| T-CVAE | 17.08 | 22.10 | 7.05 | 4.22 |
| MGCN-DP | 18.16 | 20.89 | 5.90 | 3.68 |
| CoADD | 14.53 | 21.83 | 6.99 | 4.03 |
| CoCAT | 15.08 | 24.50 | 9.09 | 5.26 |
| CoSEG | **11.45** | **26.06** | **9.80** | **5.70** |

and by 6.01/ 5.63/ 5.45/ 12.81/ 10.36 respectively in the inadequate test set. In addition, in terms of BLEU-1, **CoSEG** outperforms **MGCN-DP**, **T-CVAE**, **IE**, **HAN** and **Seq2Seq** by 1.38%/ 1.56%/ 2.2%/ 2.85%/ 2.82% respectively in the sufficient test set, and by 5.17%/ 3.96%/ 7.66%/ 8.91%/ 8.93% respectively in the inadequate test set.

2) Our **CoSEG** model has the smallest performance gap between the two test sets, which illustrates the performance of our model is not easily affected by the amount of information. Specifically, the perplexity increased by 1.55 in the inadequate test set based on the sufficient test set, and the BLEU-1 increased by 0.78%.

3) In both the sufficient and inadequate test set, the **CoSEG** model outperforms the **CoADD** and the **CoCAT** a lot, which illustrates the interaction ability of the VBF Module is much stronger than the addition and concatenation.

**Manual Evaluation** The manual evaluation results for the sufficient and inadequate test sets are shown in Table 2, where we can observe:

In both the sufficient and inadequate test set, the **CoSEG** model obtains the best coherency score and the best grammar score. Specifically, in terms of Coherency, **CoSEG** outperforms **CoCAT**, **CoADD**, **MGCN-DP**,**T-CVAE**, **IE**, **HAN** and **Seq2Seq** by 0.025/ 0.19/ 0.12/ 0.13/ 0.52/ 0.72/ 0.485 respectively in sufficient test set, and by 0.655/ 0.4/ 0.305/ 0.18/ 0.41/ 1.02/ 0.715 respectively

Table 2: Manual evaluation results of the sufficient and the inadequate test set.

| Model | Sufficient | | Inadequate | |
|---|---|---|---|---|
| | Coherency | Grammar | Coherency | Grammar |
| Seq2Seq | 1.395 | 0.655 | 0.905 | 0.780 |
| HAN | 1.160 | 0.685 | 0.600 | 0.785 |
| IE | 1.360 | 0.760 | 1.210 | 0.820 |
| T-CVAE | 1.750 | 0.785 | 1.440 | 0.815 |
| MGCN-DP | 1.760 | 0.780 | 1.315 | 0.795 |
| CoADD | 1.690 | 0.775 | 1.220 | 0.760 |
| CoCAT | 1.855 | 0.605 | 0.965 | 0.705 |
| CoSEG | **1.880** | **0.805** | **1.620** | **0.835** |

in inadequate test set. Moreover, in terms of Grammar, **CoSEG** outperforms **CoCAT**, **CoADD**, **MGCN-DP**, **T-CVAE**, **IE**, **HAN** and **Seq2Seq** by 0.2/ 0.03/ 0.025/ 0.02/ 0.045/ 0.12/ 0.15 respectively in sufficient test set, and by 0.13/ 0.075/ 0.04/ 0.02/ 0.015/ 0.05/ 0.055 respectively in inadequate test set.

Table 3: Ability to Customize Endings.

| Testset | SucR | Rationality | DiscD |
|---|---|---|---|
| Sufficient | 0.855 | 1.965 | 0.755 |
| Inadequate | 0.605 | 1.980 | 0.555 |

*Ability to Customize Endings* The ability to customize endings for different characters of our model is shown in Table 3, where we can observe:

1) In the sufficient test set, the success rate (SucR) and the discrimination degree (DiscD) are 85.5% and 75.5% respectively, which indicates that our model is able to identify the differences between characters. The SucR and DiscD in the inadequate test set are lower than there in the sufficient test set, which illustrates that the amount of character information has a certain influence on distinguishing different characters.

2) In both the sufficient and inadequate test sets, the rationality of the customized endings is close to 2.0 on the premise that the maximum score is 3.0. It indicates that our model has a high probability of 66% to predict a reasonable ending for each character.

*Combination Analysis on Product Candidates* We conduct experiments on several different combinations of product candidates. Specifically, the **CoSEG (n)** selects the product candidate generated using the $VecB_n$ breaking operation; the **CoSEG (0-256-512)** selects the product candidates generated using $VecB_0$,

Table 4: Experimental results of several different combinations.

| Model | PPL |
|---|---|
| CoSEG (0) | 12.08 |
| CoSEG (128) | 14.74 |
| CoSEG (256) | 13.06 |
| CoSEG (0-256-512) | 10.93 |
| CoSEG (0-128-256-384-512) | **9.99** |

$VecB_{256}$ and $VecB_{512}$ three breaking operations; the **CoSEG (0-128-256-384-512)** selects the product candidates generated using $VecB_0$, $VecB_{128}$, $VecB_{256}$, $VecB_{384}$ and $VecB_{512}$ five breaking operations.

The experimental results are shown in Table 4. We can observe that **CoSEG (0-128-256-384-512)** achieves the best performance. The result explains that we finally selected the product candidates generated using $VecB_0$, $VecB_{128}$, $VecB_{256}$, $VecB_{384}$ and $VecB_{512}$ five breaking operations and utilize the selected five product candidates as inputs to C-CA module. In addition, the result is attributed to the fact that **CoSEG (0-128-256-384-512)** involves more and smaller candidate which facilitate generating fine-grained semantic elements and providing more semantic combinations.

## 5   Case Study

### 5.1   Ground-Truth Endings

We present several examples of ground-truth story endings generated by baselines and our model in Table 5 to demonstrate that our model is able to generate more natural and more character-related endings than the baselines. Specifically, in the first story in Table 5, the ending generated by baseline **T-CVAE** makes the reader feel that the character is not smart enough, it has begun to *rain*, and it should not be very useful to *look at the sky* at this time; the baseline **IE** only describes *He had to go home*, without other details. And other baselines predicted wrong endings. By contrast, our model **CoSEG** allows the character to take an appropriate action, *get an umbrella*, according to the given story context, *started raining*.

In addition, the second story in Table 5 is derived from the inadequate test set. In this example, the context is a story about a wrong lamp. The baselines **HAN**, **IE** and **MGCN-DP** describe *She/We had to clean it up*, and **IE** further describes *get a new one*. Our model **CoSEG** not only allows the character to have an appropriate emotion, *I was so upset*, but also let the character take a reasonable action, *decided to get her a new one*. Obviously, the ending generated by **CoSEG** takes the character's emotions (***upset***), behaviors (***decided...get...***),

Table 5: Case Study: Endings Generated by Different Models.

| | |
|---|---|
| Context: | Ned was walking in the park one day. He noticed the sky started to turn gray. Ned turned back toward his house. He didn't quite get home before it started raining. |
| Gold: | Ned ran inside, a little wet, but happy to be home. |
| Seq2Seq: | He went back to his car and bought with his warm. |
| HAN: | He had a clean mess. |
| IE: | He had to go home and go home. |
| T-CVAE: | Ned looked around and saw the sky. |
| MGCN-DP: | he was very happy with his new car! |
| CoADD: | He decided to go to the store to buy more umbrella. |
| CoCAT: | He went outside to find his car. |
| CoSEG: | He decided to go to the store to get an umbrella. |
| Context: | We removed a lamp last week during carpet cleaning. After the cleaners left **my wife** put the lamp back. The lighting seemed much dimmer. Then **she** realized she had brought out the wrong lamp. |
| Ground truth: | I said I had not even noticed. |
| Seq2Seq: | She was so relieved. |
| HAN: | She had to clean it up. |
| IE: | She had to clean it up and get a new one. |
| T-CVAE: | I bought a new lamp . |
| MGCN-DP: | We had to clean it up. |
| CoADD: | I was so mad that she was able to get it fixed. |
| CoCAT: | She was so happy to have a new lamp. |
| **CoSEG**: | I was so **upset** , I **decided** to **get her** a new one. |
| Context: | Larry **needed a job**. He was let go from a job a month ago and hadn't found another one. His **bills were starting to mount up**. Larry was determined and applied to many jobs every day. |
| Ground truth: | He was finally hired. |
| Seq2Seq: | He was able to get a job in time and help for his job. |
| HAN: | He was able to get a job in a new city. |
| IE: | He was hired for a new job. |
| T-CVAE: | He have a job. |
| MGCN-DP: | He said he was going to be more careful. |
| CoADD: | He was able to get to work and get a job. |
| CoCAT: | He was hired at the store and bought a brand new car. |
| **CoSEG**: | He finally **got a job** offer and **paid his bills**. |

Table 6: Customizing endings for each character using our proposed CoSEG model.

| | |
|---|---|
| Context: | **I** ran and climbed over the fence. |
| | **My son** was lying in the pea gravel on the road. |
| | The car had swerved just in time. |
| | **I** raged at the **driver** for not even stopping. |
| Ground truth: | I called 911 to come get my child. |

Endings for each character:

| | |
|---|---|
| For **I**: | I ended up **falling** and I had to **go to the hospital**. |
| For **driver**: | The driver **pulled the car** and I was able to get it back. |
| For **son**: | My son was **upset**. |

| | |
|---|---|
| Context: | **I** had a dental appointment I had to go to today. |
| | While getting my teeth checked , my **dentist** told **I** had a cavity. |
| | **He** said it's probably because **I**'ve been using subpar toothpaste. |
| | **I**'ve been using the same toothpaste **he** recommended six months ago. |
| Ground truth: | Thanks a lot for the recommendation, doc. |

Endings for each character:

| | |
|---|---|
| For **I**: | I am **glad** I **have a new toothpaste**. |
| For **dentist**: | The dentist **told me** that he **had to get a new toothpaste**. |

| | |
|---|---|
| Context: | **John** was awakened by a phone call. |
| | Answering , **John** realized it was his buddy, **Rich**. |
| | **Rich** said he was stranded on a highway just outside of town. |
| | **John** drove out to pick up **Rich**. |
| Ground truth: | John drove Rich home , where they both fell asleep on the couch. |

Endings for each character:

| | |
|---|---|
| For **Rich**: | He **drove to the mall** and **bought a new car**. |
| For **John**: | John and his friends **went to the park** and had a great time. |

and relationships (*get **her** a...*) into account, which illustrates the ability of our model to obtain character's personality.

The third example in Table 5 is sampled from the sufficient test set. In this example, the context is a story about a man's bills mount up and he needs a job. Most of the baselines describe *He get a job*, as well as the ground truth and our proposed model. In addition, different from all baselines and the ground truth endings, our model further describes the man's purpose of looking for a job, ***paid bills***, which also demonstrates that our model is able to generate a more character-related ending.

## 5.2   Character-Orient Endings

In this section, we present three examples with character-orient endings (including the ground-truth endings) generated by our method in Table 6, to illustrate the ability of our model to customize endings for different characters.

In the first example in Table 6, our model customizes endings for the characters *I*, *driver* and *son*. The context is a story about a car accident that happened to a father and son. The endings generated by our model are that the *father* had to go to the hospital, the *driver* ran away, and the *son* was upset. These three endings generated by our model describe the behavior of the *father* and the *driver* and the mood of the *son*, which demonstrate the effectiveness of our proposed model to customize endings for different characters. The second example in Table 6 is a story in that a man has a cavity because he uses the toothpaste recommended by the dentist, and our model customizes endings for the characters *I* and *dentist*. The third example in Table 6 is a story in that Rich is stranded on a highway and he calls John to pick him up. Our model identifies the differences between Rich and John, generating endings, *He drove to the mall and bought a new car*, for Rich, and *John and his friends went to the park and had a great time*, for John. Rich **need a new car**, because his car is **stranded on the highway**.

## 6   Conclusion

To tackle the challenging task of customizing story endings for different characters, we propose a Character-oriented Story Ending Generator (CoSEG). Experimental results demonstrate that our proposed model can not only generate more coherent and diversified story endings compared with state-of-the-art methods but also effectively customize the ending for each character in a story.

## 7   Acknowledgements

## References

1. Chen, J., Chen, J., Yu, Z.: Incorporating structured commonsense knowledge in story completion. In: AAAI. pp. 6244–6251. AAAI Press (2019)
2. Crittenden, J.C., Trussell, R.R., Hand, D.W., Howe, K.J., Tchobanoglous, G.: Ion Exchange. John Wiley & Sons, Inc. (2012)
3. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (1). pp. 4171–4186 (2019)
4. Fan, A., Lewis, M., Dauphin, Y.N.: Hierarchical neural story generation. In: ACL. pp. 889–898 (2018)

5. Fan, A., Lewis, M., Dauphin, Y.N.: Strategies for structuring story generation. In: ACL. pp. 2650–2660 (2019)
6. Fedus, W., Goodfellow, I.J., Dai, A.M.: Maskgan: Better text generation via filling in the _ _ _ _ _ _ _ . In: ICLR (2018)
7. Feng, C., Shi, C., Hao, S., Zhang, Q., Jiang, X., Yu, D.: Hierarchical social similarity-guided model with dual-mode attention for session-based recommendation. Knowl. Based Syst. **230**, 107380 (2021)
8. Gao, J., Bi, W., Liu, X., Li, J., Shi, S.: Generating multiple diverse responses for short-text conversation. In: AAAI. pp. 6383–6390 (2019)
9. Guan, J., Huang, F., Huang, M., Zhao, Z., Zhu, X.: A knowledge-enhanced pretraining model for commonsense story generation. Trans. Assoc. Comput. Linguistics **8**, 93–108 (2020)
10. Guan, J., Wang, Y., Huang, M.: Story ending generation with incremental encoding and commonsense knowledge. In: AAAI. pp. 6473–6480 (2019)
11. Huang, Q., Mo, L., Li, P., Cai, Y., Liu, Q., Wei, J., Li, Q., Leung, H.: Story ending generation with multi-level graph convolutional networks over dependency trees. In: AAAI. pp. 13073–13081 (2021)
12. Kennedy, R.H.: Elution of uranium values from ion exchange resins (1959)
13. Kim, J., Benjamin, M.M.: Modeling a novel ion exchange process for arsenic and nitrate removal. Water Research **38**(8), 2053–2062 (2004)
14. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP. pp. 1746–1751 (2014)
15. Liu, D., Li, J., Yu, M., Huang, Z., Liu, G., Zhao, D., Yan, R.: A character-centric neural model for automated story generation. In: AAAI. pp. 1725–1732 (2020)
16. Luo, F., Dai, D., Yang, P., Liu, T., Chang, B., Sui, Z., Sun, X.: Learning to control the fine-grained sentiment for story ending generation. In: ACL. pp. 6020–6026 (2019)
17. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: EMNLP. pp. 1412–1421 (2015)
18. Mao, H.H., Majumder, B.P., McAuley, J.J., Cottrell, G.W.: Improving neural story generation by targeted common sense grounding. In: EMNLP. pp. 5987–5992 (2019)
19. Martin, L.J., Ammanabrolu, P., Wang, X., Hancock, W., Singh, S., Harrison, B., Riedl, M.O.: Event representations for automated story generation with deep neural nets. In: AAAI. pp. 868–875 (2018)
20. Mo, L., Wei, J., Huang, Q., Cai, Y., Liu, Q., Zhang, X., Li, Q.: Incorporating sentimental trend into gated mechanism based transformer network for story ending generation. Neurocomputing **453**, 453–464 (2021)
21. Mostafazadeh, N., Vanderwende, L., Yih, W., Kohli, P., Allen, J.F.: Story cloze evaluator: Vector space representation evaluation by predicting what happens next. In: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP. pp. 24–29 (2016)
22. Mou, L., Song, Y., Yan, R., Li, G., Zhang, L., Jin, Z.: Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In: COLING. pp. 3349–3358 (2016)
23. Papineni, K., Roukos, S., Ward, T., Zhu, W.: Bleu: a method for automatic evaluation of machine translation. In: ACL. pp. 311–318 (2002)
24. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–1543 (2014)
25. Schütt, K.T., Gastegger, M., Tkatchenko, A., Müller, K.R., Maurer, R.J.: Unifying machine learning and quantum chemistry – a deep neural network for molecular wavefunctions. Nature Communications (2019)

26. Segler, M.H.S., Preuss, M., Waller, M.P.: Planning chemical syntheses with deep neural networks and symbolic ai. Nature **555**(7698), 604 (2018)
27. Tu, L., Ding, X., Yu, D., Gimpel, K.: Generating diverse story continuations with controllable semantics. In: NGT@EMNLP-IJCNLP. pp. 44–58 (2019)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS. pp. 5998–6008 (2017)
29. Wang, T., Wan, X.: T-CVAE: transformer-based conditioned variational autoencoder for story completion. In: IJCAI. pp. 5233–5239 (2019)
30. Welleck, S., Brantley, K., III, H.D., Cho, K.: Non-monotonic sequential text generation. In: ICML. Proceedings of Machine Learning Research, vol. 97, pp. 6716–6726 (2019)
31. Xu, J., Ren, X., Zhang, Y., Zeng, Q., Cai, X., Sun, X.: A skeleton-based model for promoting coherence among sentences in narrative story generation. In: EMNLP. pp. 4306–4315 (2018)
32. Xu, P., Patwary, M., Shoeybi, M., Puri, R., Fung, P., Anandkumar, A., Catanzaro, B.: MEGATRON-CNTRL: controllable story generation with external knowledge using large-scale language models. In: EMNLP (1). pp. 2831–2845 (2020)
33. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A.J., Hovy, E.H.: Hierarchical attention networks for document classification. In: NAACL. pp. 1480–1489 (2016)
34. Zhang, Q., Cao, L., Shi, C., Niu, Z.: Neural time-aware sequential recommendation by jointly modeling preference dynamics and explicit feature couplings. IEEE Transactions on Neural Networks and Learning Systems pp. 1–13 (2021)