Optimizing Chinese Story Generation based on Multi-channel Word Embedding and Frequent Pattern Tree Structure

Jhe – Wei Lin\textsuperscript{a*}, Rong – Guey Chang\textsuperscript{b}

\textsuperscript{a,b}Department of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi, Taiwan, R.O.C \textsuperscript{a}ORCID 0000-0002-7922-6269

Abstract

Much space remains for generating stories in the Chinese language. In this paper, we propose a novel approach to address this issue by multi-channel word embedding and effective control of the part-of-speech structure while producing sentences to imitate the writing style. The proposed approach consists of four parts: We first preprocess the sentences to label all sentences in the data set according to the format of <SOS> <MOS> <EOS>, where <SOS> <EOS>, and <MOS> represent the beginning, the end of the sentence, and the separation between sentences. We then propose a multi-channel method to embed words by integrating traditional vectorization methods including Word2vec, Fasttext, LexVec, and Glove to improve the information in the input data. We next optimize the model architecture to effectively control the process of sentence generation based on the BERT (Bidirectional encoder representations from) model. Finally, we perform some optimization on performance. For example, Softmax function in the model was optimized to reduce the search time during training. In addition, the GAN (generative adversarial network) architecture for the data set was revised to improve the training performance of the model. All sentences in the data set are built into a tree structure and the part-of-speech structure of the next sentence was generated through model generation based on FP-tree. The experimental results show that the proposed method can effectively control the generation of Chinese stories.

Keywords: Chinese story generation method, Transformer model, Multi-channel word embedding, Data format, FP-tree, GAN.

*Corresponding author; Email: little52040234@alum.ccu.edu.tw
1. Introduction

Deep neural networks have caused revolutionary changes in many fields such as computer vision and natural language (NLP) processing. Versus computer vision, the field of the natural language still performs basic applications such as chat robots, language translation, question and answer, or reading comprehension, etc.; these applications do not actually solve the more practical sentences and semantics of natural language. Natural language generation (NLG) is a part of natural language processing and generates natural language from machine representation systems such as knowledge bases or logical forms. When this formal expression is used as a model of psychological expression, psycholinguists choose the term language output. The natural language generation system is analogous to a translator that converts data into natural language expressions. NLG has been around for a long time, but commercial NLG technology has only recently become popular. Natural language generation is the inverse of natural language understanding: Natural language understanding systems need to clarify the meaning of the input sentence to produce the machine language; the natural language generation system needs to decide how to transform the concept into language. Both text-to-text generation and data-to-text generation are examples of natural language generation.

In the NLG method survey, NLG is described as a subfield of artificial intelligence and computational linguistics, which focuses on how to build a computer system that constructs understandable English text from nonverbal information. Obviously, this definition is more suitable for data-to-text generation than text-to-text generation. In fact, (Reiter et al. 2000) focused on the former because this was the mainstream research direction at the time. Some scholars pointed out that the precise definition of NLG is quite difficult: Everyone seems to agree on what the output of the NLG system should be, but the exact input quite variable. The boundaries between different methods are also often inherently blurred. For example, text summaries can be characterized as a text-to-text application. However, many text-to-text generation methods use techniques that are also used for data-to-text. Traditionally, the NLG problem of converting input data into output text is solved by decomposing it into multiple sub-problems. Generally, these problems can be divided into the following six categories:

I. Content determination: Determine what information is included in the text under construction,
II. Text structuring: Determine the information that will be displayed in the text,
III. Sentence aggregation: Decide what information is presented in a single sentence,
IV. Lexicalization: Find the correct words and phrases to express information,
V. Referring expression generation: select words and phrases to identify domain objects,
VI. Linguistic realization: Combine all words and phrases into well-formed sentences.

Natural language processing (NLP) is a subfield of computer science and artificial intelligence that focuses on how to let computers process and analyze large amounts of natural language data. Common applications currently generate shorter sentences by analyzing longer sentences such as chat bots, automatic summarization, reading comprehension, etc. This part of the application is biased towards statistics and analysis even in the questioning and answering of reading comprehension. The answer is often only in a certain paragraph of the question, and the generated result is not expressed through sentence reorganization and analysis. Prior natural language research has had difficulties in evaluating a piece of generated data. Commonly standards such as BLEU (Papineni et al. 2002) and ROUGE (Lin et al. 2004) evaluate specific ground truth standards, but the generated results are the entire article. For an article or even a book, this evaluation standard will become unfounded because the generation of the article often hopes that it doesn’t have ground truth, which can increase the reader's sense of surprise when reading. If we send a “how are you” message to the Chabot, the expected result is that the Chabot will return "I'm fine". However, this isn’t our goal of generating story articles.

The main goal of this thesis is to train Chinese stories to achieve the expected abstract input through natural language processing and deep learning techniques and then generate article content corresponding to the abstract. In the natural language processing part, we use the Chinese Academy of Sciences word segmentation system (CKIP) to solve Chinese word segmentation question. Thus, we propose two methods for vectorization.
I. Combine Word2Vec (Mikolov et al. 2013), Global Vectors (Pennington et al. 2014) and FastText (Joulin et al. 2016), and a combination of these three vectorizations was used to propose a set of multi-channels of pretrained word-embedding methods to deal with vectorization problems.

II. The approach contains more semantic sentence expressions. The traditional sentence vectorization method will directly vectorize the processed sentence. This processing method contains less information, and thus we combine the word segmentation result with the concept of part of speech. The training data set contains the style of a certain writer when creating; this in turn contains the number of words used in each sentence and is effectively imitated and generated.

The problem of vectorization of Chinese characters is a very difficult. Method to effectively represent the smallest unit words in a vectorized manner are still in complete. This work uses the deep learning space, based on the Transformer network model as the main training architecture and combined with bidirectional encoder representation from transformers (BERT; Jacob al. 2018) to fine tune our main target story generation. The training results have a certain degree of accuracy using the generative adversarial network (GAN; Mirza al. 2014) method by discriminator and generator to enhance our usability during text generation.

2. Related Works

Here, we introduce several related works such as deep learning models, sentiment analysis, and attention mechanism. In deep learning models, we review some previous works for sentiment analysis like Word Representation, Sequence Models, and Convolutional Neural Networks. NLG can be divided into text-to-text generation and data-to-text generation. Text-to-text generation can be further divided into machine translation, summary generation, text simplification, text correction, text interpretation, question generation, etc.

In the field of machine translation, Brown (Brown al. 1993) applied statistical methods for the machine translation. They described a series of five statistical models of the translation process and gave an algorithm for estimating these model parameters for a set of mutually translated sentence pairs. The examples they gave were limited to translate between French and English, but they believed that the model could also work well on other language pairs. For common models, Och al. used statistical or heuristic models in 2003 to present and compare various methods for calculating word alignment. Bannard et al. (2018) used a bilingual parallel corpus to extract and generate paraphrases. Using the alignment technology of phrase-based on statistical machine translation, they showed how to use a phrase in another language as a pivot to identify paraphrases in one language. They defined a paraphrase probability that allows the interpretations extracted from a bilingual parallel corpus to be ranked using translation probabilities, and explained how to refine it to consider contextual information. Abstract generation is usually performed on isolated sentences regardless of the surrounding context. Clarke al. (2010) proposed a model for coherent and informative document compression. Their model was inspired by local coherence theory and formulated within the framework of integer linear programming. The experimental results showed that their model had the best performance at the time. Others (Bartoli al. 2010) published a paper on a tool that can automatically generate fake reviews of a given scientific paper. A key feature of the tool is that it is based on a small knowledge base. Of course, generating text from non-text data is also an important research direction of NLG.

The Long-short Term Memory Network (LSTM) model was inspired by the different between short-term memory and long-term memory in the human brain’s memory mechanism (i.e., a variant of the artificial neural network (Hochreiter al. 1997)). It has been used to establish a language model (LM) in 2012 (Sundermeyer al. 2012). Because of the rise of deep learning, deep learning models based on LSTM have undergone more changes.

A standard sequence-to-sequence model (seq2seq model) was proposed to develop machine translation. This model specializes in processing data whose input is a string and output is also a string. The Seq2seq model uses two LSTM model the first LSTM model encodes (Encode) the input sequence into a context vector (Context Vector), and the second LSTM model decodes (Decode) the context vector into an output sequence (Cho al. 2014; Sutskever al. 2014). This model inserts a context vector between two LSTM models. The context vector represents the semantic meaning of the input sequence. Even if the length of the model input sequence and the output sequence are different, the neural network can learn.
Many scholars have applied GAN in the field of NLP. The traditional GAN architecture cannot be directly applied due to its inability to process discrete data, and related research has stalled for three years. In 2017, Lantao et al. proposed SeqGAN (Yu et al. 2017) to use Policy Gradient to solve the problem whereby traditional GANs could not handle discrete data. Many scholars made improvements based on the SeqGAN architecture (Che et al. 2017; Lin et al. 2017; Guo et al. 2017; Zhang et al. 2017) as well as the work of Yaoming Zhu et al. In recent years, related documents on the application of GAN in the field of NLP have been published (Zhu et al. 2018), but no research on GAN has proposed to use the abstract as an input and the paper as the output model.

3. Experiment process and architecture

This paper is divided into four parts. The first part defines a novel preprocessing label format to control the model and generate sentences based on the part of speech that we provide. This format can effectively control the Chinese sentence structure generated by the model and also imitate the characteristics of a certain writer. The second is based on multi-channel word embedding: It combines different quantitative information of the model input data so that the model can contain more information advantages. Experiments showed that this method is effective. third, the feature-matrix is obtained through a method of multi-channel word embedding. This is trained through a transform architecture. The trained model undergoes generative adversarial network to generate the final GAN model that we need. The final step is story generation. Based on the data format we proposed, the input sentence is sorted into a defined format and then thrown into the model for generation. The FP-tree is used to generate the sentence structure, and the article is then finially recursively generated.

Figure 1. Proposed experimental flowchart.

4. Sentence preprocessing

This chapter will introduce several sentence processing methods: The main purpose of this thesis is to input a paragraph or sentence to generate a paragraph of text, and the style of the text focuses on Jin Yong’s famous novel "Demi-Gods and Semi-Devils"(天龍八部). Generation of the style and theme expectations are similar to the original novel. The main collection of materials is the novel "Demi-Gods and Semi-Devils"(天龍八部).
4.1. Segment the sentence between Chinese and English

The difficulty of text segmentation between Chinese and English is shown in Table 1. The example is the same sentence in Chinese and English, and the meaning of a sentence can be interpreted as “we play at the wildlife park” in English. We can easily to segment this sentence by using the space. However, in Chinese, we have different results after segmentation both results are correct but have different meanings: The former is what we want, but we cannot say that the other one wrong in Chinese because each word after the segment exists in all Chinese words. Chinese words can create new words by splitting and reorganizing. Thus, we do not have a clear rule when we use segmentation: We cannot tell the computer the meaning that we want when segmenting. This is a big problem in pre-processing of Chinese. In addition, some words cannot be segmented correctly such as the names of people, places, or some strange characters. Being able to accurately segment the sentence affects the accuracy of the data in the training data set, and there are some ways to increase the accuracy of a segment.

| Status     | Chinese sentence | English sentence |
|------------|-----------------|-----------------|
| Before segment | 我們在野生動物園玩 | We play at the wildlife park |
| After segment | 我們/在/野生/動物園/玩 | We/ play/ at/ wildlife/ park/ |

The CKIP segmentation system was used here for pre-sentence processing. Recently, the Academia Sinica of Taiwan has open-sourced the word segmentation system. This CkipTagger Chinese processing tool not only provides traditional Chinese word segmentation functions but also adds functions such as part-of-speech tagging and named entity recognition for 18 types of proper nouns. CkipTagger performed much better than Chinese stuttering when using the ASBC 4.0 Chinere corpus test set of 50,000 sentence for the Chinses word segmentation test. The accuracy of Chinese word segmentation by Academia Sinica was 97.49%. In contrast, Chinese stuttering only offered 90.51%.

4.2. Data label of Abstract and content

In the beginning, manual labeling was used to label the abstracts of 1161 chapters in the "Demi-Gods and Semi-Devils" (天龍八部) novel. This label generates longer articles with shorter sentences. This type of labeling process takes significant time. In the case of multi-person division of labor, each person's standard for abstracts is also inconsistent. This lead to difficulties in deep learning and poor results after final model training that is prone to overfitting. The data format of the data set is shown in Table 2.

| Abstract | 大理國無量山無量劍派的練武廳中，舉辦了五年一次的比武鬥劍大會，由無量劍的東、北、西三宗互相比試。此次是第九次大會。 |
| Content  | 青光閃動，一柄青鋼劍倏地刺出，指向中年漢子左肩，使劍少年不待劍招用老，腕抖劍斜，劍鋒已削向那漢子右頸。那中年漢子豎劍擋格，錚的一聲響，雙劍相擊，嗡嗡做聲，震聲未絕，雙劍劍光霍霍，已拆了三招。中年漢子長劍猛地擊落，直斬少年頂門。那少年避向右側，左手劍訣斜引，青鋼劍疾刺那漢子大腿。<newline><newline>兩人劍法迅捷，全力相搏。練武廳東邊坐着二人。上首是個四十左右的中年道姑，鐵青首臉，嘴脣緊閉。下首是個五十餘歲的老者，右手捻人的長鬚，神情甚是得意。兩人的座位相距一丈有餘，身後各站上二十餘名男女弟子。西邊一排椅子上坐東十餘位賓客。東西雙方的目光都集註於場中二人的角鬥。<newline><newline>眼見那少年與中年漢子已拆到七十餘招… |
In addition, we will add <SOS> to the front of the sentence data and <EOS> to the end of the sentence during training. This way of marking helps us effectively control the length of the generated sentence. However, we used this training method through the experiments and identified two problems.

I. The amount of data cannot be increased quickly because of the difficulties in manual labeling. It is difficult to generate long sentences from short sentences, and there is not a large amount of data to generate results.

II. The experimental results show that we cannot input sentences other than the training set as input. We try to input other sentences than the input into the model for generation, and the output result cannot be understood. The input is the data in the training set. When the model is trained to converge, we try to input the same sentence as the training set as input. The long content output by the model will be almost the same as the content in the training data set causing the model to overfit.

Because of these problems, we studied how to quickly mark up our data and how to effectively control the output conditions and generate better content. Here, we tried to control the generation of sentences instead of generating whole articles. Currently, one can only control the generated results with abstracts. This control method has no significant effect. Only through sentence control can the results converge to our expected data.

4.3. A novel preprocessing label format

Based on the data labeling in the abstract and the content in section 4.2, we could quickly process and label the data. Effectively controlling the content of the generated sentence is also very important. Based on the above two points, we propose a novel preprocessing label format. The method has good generation results in the experimental results and can effectively imitate "Demi-Gods and Semi-Devils" (天龍八部) with the faster processing of the label part. It labels through the relationship between sentences.

4.3.1. Preprocessing label with part-of-speech sentence(v1)

The original data format corresponds to the summary to the content, and the new data format is the first sentence and corresponds to the second sentence with a comma in the middle. This method can quickly process a large amount of data without manual marking. To some extent, the connections between the sentences are stronger than the abstract to the content. The first sentence is used as input to train what to generate in the second sentence. Obviously this training method can have a strong semantic structure.

The structure of the input data includes the <SOS> token and the <EOS> token. In addition to the <MOS> token, the <MOS> token is used as a token that separates the first sentence from the second sentence in our input. The complete input data can be as follows:

Input data:

\[
\text{<SOS> word}^1 \text{ PoS}^1 \text{ word}^2 \text{ PoS}^2 \text{ word}^3 \text{ PoS}^3 \text{ <MOS> PoS}^4 \text{ PoS}^5 \text{ PoS}^6 \text{ <EOS>}
\]

Label:

\[
\text{word}^4 \text{ PoS}^4 \text{ word}^5 \text{ PoS}^5 \text{ word}^6 \text{ PoS}^6
\]

From <SOS> to <MOS> represents the current sentence, <MOS> to <EOS> represents the next sentence, the word represents the sentence after segmentation, and PoS represents the word for part of speech. The current sentence contains the word and part of speech of the sentence. In the second sentence, we retain its part of speech. We hope that the model can know the part of speech of the second sentence through training.
The label part uses the word and the part of speech of the second sentence as the output. This approach has two advantages:

I. The data we input tells the model the sentence structure of the second sentence and the semantic structure of the first sentence; thus, the model can effectively generate the reference semantics and sentence structure.

II. This input design method will return to short sentences to generate long sentences. Using short sentences to generate long sentences is not very effective. This design method will improve the information available to the model during training and reduce the difficulty of generation.

4.3.2. Preprocessing label with Part-of-speech sentence(v2)

According to the experimental results in Section 4.3.1, the experimental results prove that the model can effectively imitate the part of speech of the next sentence through training and generating sentences that contain most of the part of speech. However, experimental results showed that our label contains part of speech information, and the result is that Na nouns have a high proportion in the data. The results generated by the part of speech affect the generation of semantic meaning. The original label sentence is modified into a structure that is not included in the part of speech. The purpose is to allow the model to refer to the semantic part of the content sentence during the training process that refers not only on sentence structure but also to semantic structure. Such a correction can effectively make the output focus on word generation and lead to higher correlation between words.

The new structure of the input data includes the <SOS> token, <EOS> token, and <MOS> token. The <MOS> token is used as a token that separates the first sentence from the second sentence in our input. The complete input data can be as follows:

Input data:  
\(<\text{SOS}>\text{ word}^1\text{ PoS}^1\text{ word}^2\text{ PoS}^2\text{ word}^3\text{ PoS}^3\text{ <MOS> PoS}^4\text{ PoS}^5\text{ PoS}^6\text{ <EOS}>\)

Label:  
\(\text{ word}^4\text{ word}^5\text{ word}^6\)

After this modification, the model has better results in semantic generation because there is no system for judging the quality of articles. Currently commonly used evaluation criteria are BLEU and ROGUE. Such evaluation criteria require specific ground truth, but it is difficult to have specific ground truth for article generation. Thus we cooperated with the professors of the Chinese Department of National Chung Cheng University and asked them to provide their opinions on semantics and sentences. They felt that these two parts have been greatly improved.

5. Multi-Channel word embedding

There are two popular sentiment analysis methods: learning-based and dictionary-based methods. They each have their own advantages and disadvantages. In this work, the proposed method attempts to combine them to produce a better input sentence representation. It is necessary to improve the accuracy of sentiment analysis tasks, because using multiple channels based on these two methods can complement each other and overcome each other's weaknesses. The process from the original text to the feature matrix is divided into three stages. In the first stage, the preprocessed original text will be vectorized by word embedding method. One sentence is mapped to more than 3 vector spaces. For the learning-based method, we use word2vec and GloVe to map the input sentence.

For dictionary-based dic2vec (Tissier et al. 2017), lexicon2vec (Mohammad et al. 2013) is used to represent other channels. For each type of word embedding, we have a channel, which is a matrix of shape \(T \times m\) where \(T\) is the number of tags in the input sentence, and \(m\) is \(d_1\), \(d_2\), and \(d_3\) as shown in Figure 2. Next, we used a \(l \times m\) filter to slide across three channels. This layer was used as an auto-encoder layer or feature engineering. The output size is determined by the number of filters used in each channel. If two filters are used per channel, then there are six filters. In the final stage, each \(l \times m\) filter will output a vector \(T \times 1\). The final feature matrix is generated by merging all vectors. In our example, the shape of the feature matrix should be \(T \times 6\).
6. Model architecture

The model architecture is divided into two parts: The first is our pretrain model, and the second uses Generative Adversarial Network (GAN) to train the model twice to enhance the model’s ability to generate content. We rewrite the BERT model proposed by Google. The input of the word vector uses the same method as Google and then the Positional Encoding is used to add a unique vector to each position in the sequence.

6.1. The pretrain model

Each input of the model must be subjected to positional encoding because the self-attention mechanism does not consider location information. That is, a unique value is added to each word vector for different positions. The following sine and cosine functions of different frequencies are used to generate unique value, where \( \text{pos} \) represents the position of the word, and \( i \) represents the dimension of the word.

\[
\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i/d}}\right) \quad (1)
\]

\[
\text{PE}(\text{pos}, 2i + 1) = \cos\left(\frac{\text{pos}}{10000^{2i/d}}\right) \quad (2)
\]

As shown in Figure 3, the future matrix generated by data preprocessing is used as the input, and position coding is then performed. We then enter the two-way attention mechanism, and then use the output of the two-way attention mechanism to generate a vector representation of the output text. Finally, we use linear transformation to decode these output text vectors and use the Softmax function to determine which words in the probability distribution match the meaning of the abstract.

However, the vocabulary of a novel usually has numerous words, and it is time consuming to use the traditional Softmax function to compute the rate distribution as shown in Equation 3.
Here, where \( K \) is the total vocabulary of the lexicon, and \( z \) is the input vector of the Softmax function. In the Softmax function, the exponent is used to convert the value into a positive number, and the values are then added in the denominator. The final output vector can be expressed as \( \sigma = [\sigma_1, \sigma_2, \sigma_3, ..., \sigma_n] \) for any integer \( l \), \( 0 < \sigma_i < 1 \) and \( \sum_{i=1}^{K} \sigma_i = 1 \). Consequently, \( \sigma \) can be regarded as a probability distribution. The index of the maximum value is the vocabulary that is generated by the story network in this distribution. However, it takes a considerable amount of time when the Softmax function is calculated each time because the total vocabulary \( K \) is very large. Inspired by the Ziff’s law (Lestrade et al. 2017), we applied the adaptive Softmax function to the story generation network instead of using the traditional Softmax function. According to Ziff’s law, the frequency of each vocabulary is not the same. The adaptive Softmax function uses the same concept to find the words of vocabulary \( K \) with the highest frequency. First, we calculate the Softmax function with \( K' \) to see if the maximum value of the probability distribution of the result falls within \( K' \). The process of the adaptive Softmax is shown in Figure 4 to calculate the less frequent \( K-K' \) vocabulary.

![Figure 4. The calculation process of the adaptive Softmax function.](image)

In order to reduce the redundant generation of repeated sequences in the story generation network, we use the label smoothing method to modify the cross entropy to replace the traditional cross entropy as the Loss function. First, we modify the real label distribution \( \lambda\rho \) of the data to the label smoothing distribution \( p \) in Equation 4, where \( 0 \leq \epsilon \leq 1 \).

\[
p_i = \begin{cases} 
1 - \epsilon & \text{if } i = y \\
\frac{\epsilon}{K-1} & \text{if } i \neq y 
\end{cases} \quad (4)
\]

The cross entropy is used as the loss function of the story generation network distribution \( q \) and the smooth-label-distribution \( p \). The goal of the overall training story neural network is to minimize the loss function, as shown in Equation 5.

\[
\text{loss} = -\sum_{i=1}^{K} p_i \log q_i \quad (5)
\]

### 6.2. Generative adversarial network for pretrain model

We apply the story generation network in a generative adversarial network for training to generate more real-world data distribution. The traditional Generative Adversarial network is composed of a generator and a discriminator. The generator constantly generates a sequence to fool the discriminator and the discriminator constantly recognizes the sequence produced by the generator. The traditional objective function of generating a counter against the network can be expressed as in Equation 6 where \( G \) represents the generator, \( D \) represents the discriminator, \( p_z \) is a random distribution, and \( p_{data} \) is the distribution of real-world data. The objective is to minimize the function of the maximized discriminator.

\[
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (6)
\]
We use the story generation network as the generator and the text convolution as the discriminator. However, the input of the story generation is a sequence of randomly distributed digests, the objective function can be rewritten as in Equation 7 where $p_{\text{text}}$ is the distribution of the original text data, $p_{\text{summary}}$ is the distribution of the summary data, and the overall architecture of the GAN is generated as shown below.

$$
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{text}}}(x) [\log D(x)] + \mathbb{E}_{x \sim p_{\text{summary}}}(x) [\log D(G(x))] \\
+ \mathbb{E}_{x \sim p_{\text{summary}}}(x) [\log (1 - D(G(x)))]
$$

(7)

![Figure 5. The Generative Adversarial Network for pretrain model](image)

**7. Generation flow**

The model generation is divided into two parts: The first is the sentence input through the CKIP tool to obtain the segmentation result and part of speech. Second, we use the label with the part-of-speech we proposed to splice the sentence into the input format and then throw it into the model generation. This part follows the candidate rules for the next sentence. According to the input rules we defined $<\text{SOS}> <\text{MOS}> <\text{EOS}>$, the generated result of the first part will be used as the next sentence input from $<\text{SOS}>$ to $<\text{MOS}>$. From $<\text{MOS}>$ to $<\text{EOS}>$, we will perform FP-tree analysis to find the most suitable grammatical rule to fill in the sentence rule as the next sentence. This process is repeated to generate a complete content. The complete experimental process refers to Figure 6 below.

![Figure 6. The sentence generation flowchart.](image)
7.1. Create FP-tree of data set

Here, we use FP-tree (Han et al. 2000) to match the part of speech of the next sentence. The following section introduces the process of building the data set using FT-Tree and how to choose the part of speech for the next sentence. The following Figure 7 shows the FP-tree creation template. The creation of the FP tree first needs to rely on the creation of the item Header Table. We first, scan our original data to find out the frequency of each unit and create the Header Table. We then use the original data and sort the data according to the size of the unit in the header table. We then use the header table and the sorted sentence list to create a FP-tree.

![Figure 7. The FP-tree in the proposed data format](image)

7.1.1 The candidate sentence screening from FP-tree

The method of finding candidate words first judges whether all parts of speech between <MOS> and <EOS> belong to the high-frequency part of speech. If this step is based on the proportion of each part of speech, we calculate a probability and assign to it. If not, we do not give the word probability and omit it, and finally generate the final candidate word according to each probability as the root to find the sentence structure in the FP-tree. The detailed process can refer to Figure 8.

![Figure 8. The candidate search](image)

After obtaining the candidate words, we return to the FP-tree to find the prefix and postfix. The starting point is to look for prefix and postfix because the same part of speech in the FP-tree will not only appear in one tree. The current method used the highest pointer value as the starting point. The final part-of-speech sentence is generated until all the prefixes are found. The part of speech in the FP-tree is not the part of speech of the real sentence, and the order of the sentence part of speech will be disrupted when building the FP-tree. We cannot use the part of speech of the final sentence directly. We also created a csv file that includes all sentence parts-of-speech structures in the training data set. The part-of-speech sentence is compared to the original part-of-speech sentence using FT-tree and finally the part-of-speech structure of the original sentence is found. The detailed process is shown in Figure 9.
We can obtain the next sentence through the GAN model, and use the part of speech between <MOS> and <EOS> through the CSV created by FP-tree. The sentence part-of-speech list is used to generate the next sentence’s part-of-speech arrangement. Through the recursive method, we can quickly generate as many sentences as we need in an article.

8. Experiment Results

In this chapter, we will show the experimental results of each part. First, we use: BLEU to propose semantic understanding indicators and quantify the degree of network understanding semantics generated by the story. Second using multi-channel of word embedding to improve the semantic recognition effect experiment. The final is sentence result generated by our model.

8.1. Semantic understanding indicators to quantify

Here, we control the degree randomness of the sequence GAN by controlling the input story. This and it can lead to controlling the degree of semantic information contained in the sequence. We use “John is very honest but he has cleanliness” shown in Table 2 as an example that uses English to represent the result of Chinese words. The word order is fixed but the words are randomly sorted, which is called "partial random". The vocabulary that is randomly sorted is called "completely random".

| Status            | Original sentence          | Partial random                     | Completely random                  |
|-------------------|---------------------------|------------------------------------|-----------------------------------|
| Vocabulary        | John is very honest but he has cleanliness | John is very cleanliness but he has honest | Isvery John cleanliness has he honest |
| Part of speech    | N. Adv. Adj. Conj. N. V. N. | N. Adv. Adj. Conj. N. V. N. | N. N. Adv. V. Adj. Conj. N. |
| Level of information | High                   | Medium                             | Low                                |

Therefore, this research proposes a semantic understanding index to quantify the degree of semantic understanding of the story generation network, as shown in Equation 8.

\[
\text{semantic understanding indicators} = \frac{\text{Original BLEU} - \text{complete random BLEU}}{\text{complete random BLEU}} \quad (8)
\]

In addition, Equation (9) represents the grammatical understanding indicator.

\[
\text{grammatical understanding indicators} = \frac{\text{partial random BLEU} - \text{complete random BLEU}}{\text{complete random BLEU}} \quad (9)
\]
We design an experimental process to validate the story generation network. First, the summary of the “original text”, "partially random”, and "completely random" are respectively input into the story generation network. We then calculate the semantic understanding of indicators and grammatical understanding of indicators.

| Training set | Original sentence | Partial random | Completely random |
|--------------|-------------------|----------------|------------------|
| Pretrain model | 0.11153 | 0.0506 | 0.04805 |
| GAN model | 0.09746 | 0.04386 | 0.03743 |

| Training set | Semantic understanding indicators | Grammatical understanding indicators |
|--------------|-----------------------------------|-------------------------------------|
| Pretrain model | 132% | 5% |
| GAN model | 162% | 7% |

| Training set | Semantic understanding indicators | Grammatical understanding indicators |
|--------------|-----------------------------------|-------------------------------------|
| Pretrain model | 0.03523 | 0.03467 | 0.0368 |
| GAN model | 0.02979 | 0.02897 | 0.02831 |

| Training set | Semantic understanding indicators | Grammatical understanding indicators |
|--------------|-----------------------------------|-------------------------------------|
| Pretrain model | 7% | 3% |
| GAN model | 20% | 14% |

Tables 3 and 5 show the experimental results of the two methods for the training set and the test set. The BLEU of the pretrain mode is greater than the BLEU of GAN model according to Tables 4 and 6, respectively. The semantic understanding indicators of the GAN model is larger than the semantic understanding indicators of the pretrain model, and the grammatical understanding indicators of the GAN model are greater than the grammatical understanding indicators of the pretrain model. As a result, the pretrain model contains only the information that has been read; the GAN model seems to really understand semantics and grammar.

8.2. The positive and negative comment results with Multi-Channel of Word Embedding

We will use the model framework CNN and LSTM to determine whether the article comments are positive or negative. We will then detail the environment in our experiment and the test data set used including CNN filter, batch, epoch. And other information. The final experiment evaluates accuracy and loss value during the learning process. This experiment uses two data sets including a Twitter data set and a movie reviews data set. The data set description is shown in the Table 7.

| Datasets | Total | Description |
|----------|-------|-------------|
| Twitter | 1578.61 k | The Twitter dataset has 790.178 positive and 788.436 negative tweets. Max sequence length is 175 words. |
| Movie Reviews | 10.662 k | The movie dataset has equal positive and negative sentence. Each sentence is a review with a max length of 56 words. |

The input data is preprocessed before entering the model. The preprocessing stage includes removing emoji, mentions, websites, strange Unicode and symbols, converting words into indexes, and building vocabulary dictionaries. We implemented the proposed method to perform various experiments consisting of different models on two data sets. We use Tensorflow to train deep learning models. Table 8 shows all the settings for each model.
Table 8. CNN model configuration

| Models   | Channels | Filters           | Batch Size | Epoch |
|----------|----------|-------------------|------------|-------|
| CNN-1    | 1        | [64,128,256, 64]  | 256        | 100   |
| CNN-2    | 2        | [64,128,256, 64]  | 256        | 100   |
| CNN-3    | 3        | [64,128,256, 64]  | 256        | 100   |

Table 9. Results with each dataset and type of word embeddings

| Exps       | Twitter (Acc/Loss) | MR(Acc/Loss) | Type                           |
|------------|--------------------|--------------|--------------------------------|
| CNN-1      | 66,8/0,629         |              | GloVe.50                       |
| CNN-1      | 70,2/0,600         |              | GloVe.300                      |
| CNN-1      | 66,8/0,623         |              | Word2Vec.300                   |
| CNN-1      | 69,7/0,572         |              | LexVec.300                     |
| CNN-2      | 72,1/0,563         |              | Glove.300+LexVec.300           |
| CNN-2      | 66,4/0,623         |              | Word2Vec.300+LexVec.300        |
| CNN-2      | 67,8/0,609         |              | Word2Vec.300+GloVe.300         |
| CNN-3      | 75,0/0,577         |              | Word2Vec.300+Lex2Vec.300+dect2Vec.100 |
| Our Method +CNN-2 | 81.25/0.4347       | >95 %        | Glove.50+LexVec.100           |
| LSTM-2     | 81.25/0.442        | >85%         | Glove.50+LexVec.100           |
| Our Method +LSTM-2 | 85.35/0.3406       | >98%         | Glove.50+LexVec.100           |

Figure 9. The accuracy/loss with CNN and two channels of pre-trained word embedded for Twitter dataset.

Figure 9 is the entire experimental result of our models combined with CNN and two channels of pre-trained word embedding for the Twitter dataset. The figure represents the accuracy and loss of the models during the training step. These metrics can be found through experiments whether in accuracy or loss. Through training the model lead to low loss and high accuracy, and this does not happened with the overfitting.

Figure 10. The accuracy/loss with LSTM and two channels of pre-trained word embedding for Twitter dataset.

Figure 10 shows the result of our model combined with LSTM and two channels of pre-trained word embedding for the Twitter dataset. These data also represent the accuracy and loss of the models during the training step. The loss is low and the accuracy is high without overfitting. The accuracy of LSTM is higher. In contrast, a traditional RNN model has a lower understanding of the longer strings than LSTM.
8.3. Content from model generation

The above architecture can construct the model to generate the final content. The experimental results include the training loss and testing loss of the model during the training process as well as the experimental environment and the generated results under different epochs. A server with a 64-bit processor Intel® Core™ i7-9700 CPU @ 4.7 GHz had 8 physical cores/128 GB/2x RTX 3090 24GB SLI/Ubuntu 18.04. This method was implemented with Python programming language version 3.5; anaconda was used to construct a virtual environment, and the model framework is written by Pytorch.

| Datasets                          | Total | Description                                                                 |
|----------------------------------|-------|-----------------------------------------------------------------------------|
| "Demi-Gods and Semi-Devils"(天龍八部) | 29775 | The data set contains all the contents of the 50 chapters of "Demi-Gods and Semi-Devils"(天龍八部), which are divided into sentences by punctuation |

| Input sentence                                                                 | Output Content                                                                 |
|------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| 断誉愛王語嫣，但鍾靈對他情深義重，願意幫斷公子找解藥。 | 菊峯喝道：我武功高強，盧竹你我意氣相投，曾經在寮國與壞人惡鬥。 |
| 王語嫣一心一意；公子於他不理，又是因為他自己徒兒的武功，當初有限，說得便為再一起，沒想到這三中年人丫頭，胡說八道，一布親眼不見個凌波微步。 | 喬峯不敢再語，可不敢說甚麼參與。 |
| 他跟著過即走，抱住了我面幕，苦笑說到：好解，他廝！但斷譽關心白世靖夫人和阿碧先生雖想必，但對手上穴之勁力甚是危險，我又有砍，也能買人答應。 | 阿朱心上驚，大聲心道：我這原是上前。 |
| 行出數裏，忽見閣樓中傷口下上前一個女子，模樣是西夏高僧。 | 喬峯見了永遠一人行事之手段，知道想來船極凌厲的餘地，雖未身分之訊息，都查問自走近，欲便給那許多身中話出言低，將來便將人自己不易細細。 |
| 喬峯出去問，喬峯水大手下一轉念：喬兄弟，喬大師！一座大漢在人中一班，從那樹上出去。 | 喬峯背上的事，說道：這人惡人老弟子，快在我後，說，也不沒有我。 |

Part of speech accuracy 63% 58%

| Input sentence                                                                 | Output Content                                                                 |
|------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| 断誉愛王語嫣，但鍾靈對他情深義重，願意幫斷公子找解藥。 | 菊峯喝道：我武功高強，盧竹你我意氣相投，曾經在寮國與壞人惡鬥。 |
| 界人見了這幾口人，已跟王語嫣等給副長的鬍子牽了幾層，也之甚是危険，但他將這一顆中年為毫不，卻是這樣的掌力。 | 菊峯喝道：我武功高強，盧竹你我意氣相投，曾經在寮國與壞人惡鬥。 |
| 王語嫣和他說手胡說八道，只知他徒兒既重傷，內力傷人，妙極之極，雖然隔餘地，但被他刺的不停，可是用手指和契丹人地趕來道，又無重傷，便說到：是殺他不易，走出近步，說道你自罪，無愧，成何伯父？ | 阿朱一見他被他抱著爹娘的痛苦，忽然死的力氣快，聽他殺了口，用在他鼻上清清睜，十分好笑，喝道：喂，他叫他騙我來啦！菊花道：是！菊峯既慢慢看，一隻也知撞在下新，知道他武功，這又人不行，不便，好焦急。 |
| 木婉清奇道：要是慢！斷譽力氣大吃一驚，問道怎麼沒有？木婉清道：爹爹！包三臉上一如何忙，叫道：好的？王語嫣道：這人是窮凶惡極的傷，脫身，窗中抗拒，好生欽佩。 | 他心中臉一走不，胸口自極奇怪，前來跪下的手，模樣也大來，縱身搖頭阿朱道：為走了出，立時去去還了人。 |

Part of speech accuracy 67% 65%
The following table shows the actual generated results with different epoch from 1000 and 1500. The model can be seen in part of the content where we learned about the relationship of the input sentences; the sentences of some sentences are still incoherent. This may be because the content of the novel "Demi-Gods and Semi-Devils" (天龍八部) contains "Classical Chinese" content, which leads to some incorrect word segmentation results. Some words in classical Chinese are also used in a stricter context under certain circumstances. They are not as flexible as the vernacular. Versus previous versions, there is a substantial improvement whether in semantic or semantic improvement. The composition of vocabulary is more logical and is generated according to our defined label format. The part of speech accuracy in the experiment represents that each piece of data in the defined label format contains the current sentence and a part of speech form the next sentence to make a combination, i.e., the sentence can obtain the part of speech structure of the next sentence during training and according to this part of speech architecture. Accuracy describes the reliability of the sentence structure generated by this article and the sentence structure given during training. The results show that the part-of-speech accuracy of the generated results gradually improves as the number of epoch increases; this means that the model can learn the rules of the sentence.

5. Conclusion

We propose a method based on Multi-Channel of Word Embedding that enables the model to receive more sentence information and effectively improve the model’s cognition of semantics. Through the experimental results, we see that using Glove and LexVec can achieve the best training model. In the model architecture part, we use the architecture proposed based on BERT and use the multi-layer Multi-Head Attention Layers as the main architecture inside the model. We also use the positional encoder to strengthen the importance of each word position in the sentence. Softmax is optimized to improve the training efficiency in the model training process to reduce the computing time. This was combined with the generative adversarial network (GAN) architecture to strengthen the pretrain model for sentence correctness. Finally, we use the preprocessing label format we proposed in the model generation part to process the input data; thus, the input information contains semantic and sentence data combined with FP-tree to select candidate sentence structures and use this structure for content. Chinese story text generation is still an open area to be developed including the interpretability of the deep learning model and the rationality of evaluating the quality of the text. If the machine can understand the semantics, then it can even learn "fidelity, readability, intelligence" in the text.

Acknowledgement: The authors would like to thank all colleagues and students who contributed to this study. We are grateful to Pro.Chang for her collaboration during preliminary investigations. We would like to thank anonymous reviewers and the editor for their comments.

Funding Statement: None.

Conflicts of Interest: The authors have no conflicts of interest to report.

References

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. Natural Language Engineering, 3(1), 57-87.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).

Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In Text Summarization Branches Out (pp. 74-81).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on Empirical Methods in natural language processing (EMNLP) (pp. 1532-1543).
Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. Computational linguistics, 19(2), 263-311.

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. Computational Linguistics, 29(1), 19-51.

Bannard, C., & Callison-Burch, C. (2005, June). Paraphrasing with bilingual parallel corpora. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05) (pp. 597-604).

Clarke, J., & Lapata, M. (2010). Discourse constraints for document compression. Computational Linguistics, 36(3), 411-441.

Bartoli, A., De Lorenzo, A., Medvet, E., & Tarlao, F. (2016, August). Your paper has been accepted, rejected, or whatever: Automatic generation of scientific paper reviews. In International Conference on Availability, Reliability, and Security (pp. 19-28). Springer, Cham.

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

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. In Thirteenth Annual Conference of the international speech communication association.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. arXiv preprint arXiv:1409.3215.

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017, February). Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the AAAI conference on artificial intelligence (Vol. 31, No. 1).

Che, T., Li, Y., Zhang, R., Hjelm, R. D., Li, W., Song, Y., & Bengio, Y. (2017). Maximum-likelihood augmented discrete generative adversarial networks. arXiv preprint arXiv:1702.07983.

Lin, K., Li, D., He, X., Zhang, Z., & Sun, M. T. (2017). Adversarial ranking for language generation. arXiv preprint arXiv:1705.11001.

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., & Wang, J. (2018, April). Long text generation via adversarial training with leaked information. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).

Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., & Carin, L. (2017, July). Adversarial feature matching for text generation. In International Conference on Machine Learning (pp. 4006-4015). PMLR.

Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., & Yu, Y. (2018, June). Texygen: A benchmarking platform for text generation models. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (pp. 1097-1100).
Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. ACM sigmod record, 29(2), 1-12.