# Novel Writing by means of Artificial Intelligence

**Hyung-Kwon Ko** HYUNGKWONKO@GMAIL.COM
Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul, 04763 Korea

## Abstract

*The novel has been a quintessential beauty of the human mind for centuries. It has been evaluated as something that only human beings can create but news is coming forward in 2018 that artificial intelligence is creating sound works such as poems and novels. Contrary to the current atmosphere of being at the center of attention in the marketplace, the stories regarding this topic is quite hard to find in Korea. In this paper we propose a novel writing model that uses Bidirectional LSTM and Doc2Vec which can help generate Korean short stories with a sentence recommender system. The sentence tagging job was taken into account during the data preprocessing that results in more natural flow in comparison to the model without it as well as the temperature function being implemented internally that enables the model to select diverse sentences.*

## 1. Introduction

The most popular frameworks for handling phonic, textual data sets are Recursive Neural Network(RNN) and Long-Short Term Memory(LSTM) [7, 13, 21]. Since a novel has plot, the information that was used in the beginning needs to be remembered and reused from time to time until the very last page for natural context. From this perspective, RNN can decrease the model performance because of the vanishing gradient problem [3]. Instead LSTM uses two different memory routes, that can be trained to generate better outcome from the combination of its long term and short term memory. It has many variants [4, 6] but especially BiLSTM is an enhanced version of LSTM that is being trained bi-directionally which is known to be better at making results whatever it is aiming at and that is the reason why it is used in many current studies [5, 9].

Another important aspect that we had to care about is what unit of input data we should use. A novel is made up of many scenes and the scenes are a group of sentences. Likewise sentences are a group of many morphemes. We can see there are many candidates of units in training LSTM models and choosing the right one can have an enormous effect on almost everything of this project. We decided to make sentence to sentence prediction models so as to avoid grammatical errors in single sentences. In addition, we used Doc2Vec in this process to vectorize all sentences in a training dataset into a set of numbers.

The dataset we used is a fan-fiction - a short story usually made by fans of boy groups in Korea such as *DongBangShinKi* or *BigBang*. The stories have a lot of characteristics in common with web-novels which are one of the popular forms of short stories these days. For example, they consist of many dialogues and the plot flows very quickly unlike traditional novels. Moreover, fan-fictions are easy to access and download since it is wide-spread in online community. It can be reproduced without any permission because the writers never appear on the surface. To be more specific, all fan-fictions contain some inappropriate fake stories about a member of the boy group which naturally leads the writer possibly of being sued from the boy group's company. There are no writers who are willing to reveal their identity. As a result fan-fiction is totally free from copyright. The amount of data we gathered is about 300MB but only 300KB were used in the end because much time was required to preprocess it.

Using these concepts and datasets, 5 models were built and trained to recommend the next sentence. A person can choose one sentence among what the A.I. offers and then the same process is repeated for user-given times. Lastly the concatenated sentences compose the scene and if the scenes are connected, it becomes a novel.

*Proceedings of the $30^{th}$ International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. 

## 2. Related works

Deep learning has many subgroups like RNN and CNN. We used one of the popular deep learning architecture, Bidirectional LSTM, building our model. Deep learning is showing its powerful performance in many fields and most top-level companies like Google, PayPal, Netflix are creating values by adopting deep learning based models.

This work also stretches from natural language understanding(NLU) to natural language generation(NLG). They should be treated in a different manner but are highly correlated and belong to one big group so called natural language processing(NLP). In terms of NLU, the computer must pick up on the meaning of each vector as a Korean sentence so it can catch the pattern or flow how the novel goes on. NLG deals with more outcome-related part which is generating the novel. Thus, NLG is possible only when the NLU part for machine is done seamlessly.

**The day a computer writes a novel by Japan's Future University Hakodate(2016):** As mentioned, a few precedent works on this topic are publicly announced. The most well-known one is 'The day a computer writes a novel' by Japan's Future University Hakodate in 2016 that drew a huge attention to the public by passing the first screening of 'The Nikkei Hoshi Shinichi Literary Award'. Though there are some negative properties to be pointed out. Firstly, the limitation of dialogue exists in that the conversational patterns are stuck in a few types, besides, the emotional interchange that people might expect from an ordinary novel does not appear as it contains many monologues. Finally and most importantly, as the professor who wrote this novel have admitted, there were too many human labors involved that it is hard to be deemed as pure work of artificial intelligence [12].

**Sunspring by NYU A.I. researcher R. Goodwin(2016):** On the same year, NYU A.I. researcher, R. Goodwin, co-worked with a filmmaker, O. Sharp, and produced a science fiction short film. Critics complimented on this work, while acknowledging that it had somewhat an awkward and bizarre way of storytelling at the same time.

**Shelley by MIT(2017):** MIT added a record in this field by introducing 'Shelley' that writes horror stories on Twitter. She was named after the mother of Frankenstein, M. Shelley and was raised to read eerie stories. She is a deep learning based A.I. and the writing process is pretty much the same as the other works. She randomly picks a snippet of a text and writes creepy stories using its creative mind. However, due to her technological limitation, she returns a better outcome when working with human by taking turns.

What mentioned above are breakthroughs in A.I. based creation but in reality all of them are criticized because they are not solely based on artificial intelligence but many parts were supported by people who made them. Mass media companies encouraged this atmosphere by overstating what the researchers have done with sensational phrases as if A.I. can write an intact novel by itself. However there is indeed a plenty of room for improvement.

## 3. Long Short Term Memory

RNN is known for its bad performance when related information and the application of it is long in distance because of the vanishing gradient problem [3].

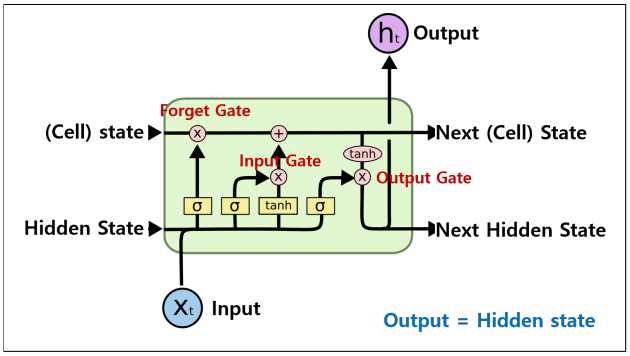


*Figure 1.* LSTM structure [15]

LSTM was introduced in [8] to solve this long term dependency. It consists of four main gates - input gate($i_t$), forget gate($f_t$), update cell state($C_t$) and output gate($o_t$) - which work interactively to decide what information ought to be stored or erased.

**Forward Pass:** The detailed computation of forward pass is given by,

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$
$$\tilde{C_t} = tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t} \quad (4)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$
$$h_t = o_t * tanh(C_t) \quad (6)$$

where $b_t$ and $W_t$ mean the bias and weight matrix on gate $t$ when $x_t$ is the input and $h_t$ is the hidden vector. The logistic sigmoid function and hyperbolic tangent function are used as an activation function and noted as $\sigma$ and $tanh$.

**Backpropagation Through Time(BPTT):** The weights of the model is tuned meticulously as the training goes back and forth while fitting into given problem. The BPTT of LSTM is computed by,

$$\delta h_t = \Delta t + \Delta h_t \tag{7}$$

$$\delta C_t = \delta h_t \odot h_t \odot (1 - tanh^2(C_t)) + \delta h_{t+1} \odot f_{t-1} \tag{8}$$

$$\delta \tilde{C}_t = \delta C_t \odot i_t \odot (1 - \tilde{C}_t{}^2) \tag{9}$$

$$\delta i_t = \delta C_t \odot \tilde{C}_t \odot i_t \odot (1 - i_t) \tag{10}$$

$$\delta f_t = \delta C_t \odot C_{t-1} \odot f_t \odot (1 - f_t) \tag{11}$$

$$\delta o_t = \delta h_t \odot tanh(C_{t-1}) \odot o_t \odot (1 - o_t) \tag{12}$$

$$\delta x_t = W^T \cdot \delta gates_t \tag{13}$$

$$\Delta h_{t-1} = U^T \cdot \delta gates_t. \tag{14}$$

With these calculations, the computation for the update of each weight matrix is given by,

$$\delta W = \sum_{t=0}^{T} \delta gates_t \otimes x_t \tag{15}$$

$$\delta U = \sum_{t=0}^{T} \delta gates_{t+1} \otimes h_t \tag{16}$$

$$\delta b = \sum_{t=0}^{T} \delta gates_{t+1} \tag{17}$$

where $\Delta t$ is the output difference by the other layers, $\Delta h_t$ is the output difference by the next time step of LSTM, $W$ is the weight matrix of hidden state($h$), and $U$ is the weight matrix of input($x$).

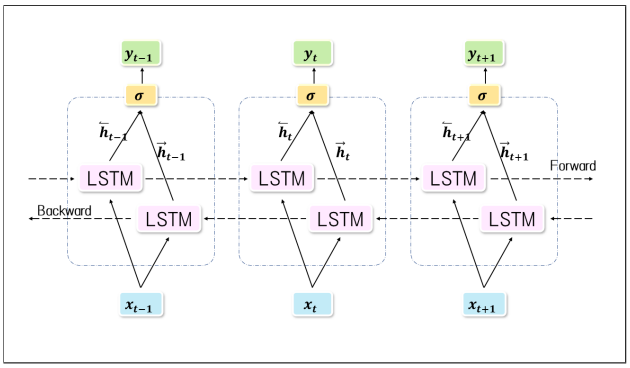


*Figure 2.* Bidirectional LSTM architecture

**Bidirectional LSTM:** Bidirectional RNN was first proposed in [20] and then it was modified for the LSTM. It is trained using both past and future information whereas the Vanila LSTM only uses past information. It is known to yield better outcome when it is properly used. With the fixed constant $n$, $\overrightarrow{h_t}$ is calculated using inputs in a forward sequence from $T-n$ time to $T-1$. On the other hand $\overleftarrow{h_t}$ is calculated using the inputs in reverse order from $T-n$ time to $T-1$. Then the two of them are merged with the calculation below,

$$y_t = \sigma(\overrightarrow{h_t}, \overleftarrow{h_t}) \tag{18}$$

where $\sigma$ function is used to combine the two output sequences. It can be chosen as any kind of function by the person who build it. Usually a concatenating function, average function, multiplication function and summation function is used.

## 4. Word Embedding

The previous approach of word embedding includes one-hot encoding however this was not suitable to capture the similarities and differences among the training words as well as it spent a lot of space because of its sparse characteristic. As a new methodology, NNLM was proposed [2] in early 2000 but took too much time in training session. Though a few years later, Word2Vec was able to solve this by introducing forward and backward propagation that makes the model structure more simplified, which results in an outstanding performance [14]. Word2Vec is known to have two main advantages, memory efficiency and the ability to capture the relationships among the training wordss.

### 4.1. Word2Vec

Word2Vec provides an efficient way for computing vector representation of words with CBoW and skip-gram method. CBoW is trained to predict the center word from the surroundings while skip-gram predicts surrounding words using the center one. Both of them are known for their good performace. But Skip-gram is more popular to researchers because CBoW requires more training time.

When a sequence of training words $w_1, w_2, \cdots, w_T$ were given, the objective of skip-gram model is to maximize the average log probability

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \le j \le c, j \ne 0} \log\, p(w_{t+j}|w_t) \tag{19}$$

where $c$ is the size of the training context(window size). $\log\, p(w_{t+j}|w_t)$ is defined using the softmax function

$$\log\, p(w_O|w_I) = \frac{exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W} exp({v'_w}^T v_{w_I})} \tag{20}$$

where $v_w$ and $v'_w$ are the "input" and "output" vector

representations of $w$, where $W$ is the number of words in the vocabulary.

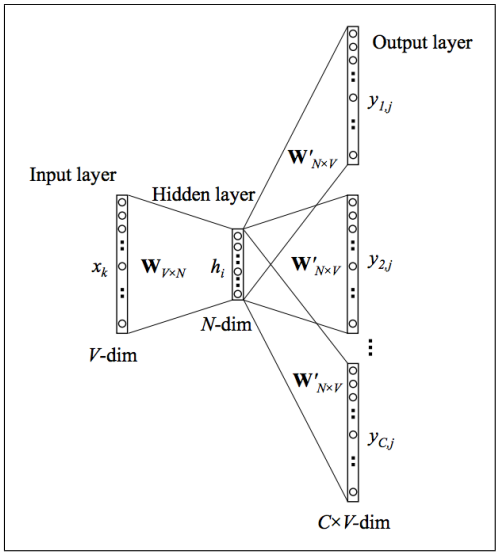


*Figure 3.* Skip-gram training architecture [19]

### 4.2. Doc2Vec

Doc2Vec was proposed a year after Word2Vec had become public [11]. Doc2Vec represents a paragraph to a set of vectors that makes the computer to get a grip on the meaning of each sentence. The paragraph vectors can be trained by two methods - distributed memory and distributed bag of words - and uniquely determined with the words in it. On the other hand, the word vectors are shared through the entire dataset. For example the word 'apple' used at first sentence has the same vector representation of that used at another sentence. In the model, the concatenation or average

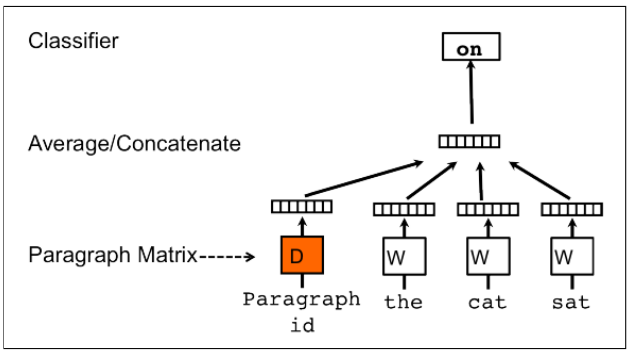


*Figure 4.* Distributed memory model architecture [11]

of word vectors is used to predict the next word. The paragraph vector represents the missing information from the present context and can work as an additional memory of the topic.

## 5. Data Preprocessing

Data preprocessing is the most important work for the best outcome. The work we have done is divided into 3 main parts which are PoS tagging, minimizing the number of named entities and sentence tagging.

### 5.1. PoS tagging

The biggest characteristic of Korean language is that it is an agglutinative language which requires to be analyzed by its morphemes thoroughly so that the computer can catch the root meaning of each word in a sentence. It consists of 9 particles officially but this is not enough to scrutinize the details. The Kkma class in KoNLPy package helps tagging words with its part of speech with 56 small groups [16]. Some of the instances are NNP(proper noun), NP(pronoun), OL(foreign language) and so on. This work is important in two perspectives.

| Before PoS Tagging | After Pos Tagging |
|---|---|
| 사람은 | '사람/NNG', '은/JX' |
| 은목걸이 | '은/NNG', '목걸이/NNG' |

*Table 1.* Distinguish homonym with PoS tagging

Firstly, there are homonyms that have to be distinguished. We can tell from the given example that the computer is properly getting the homonym such as '은' can be interpreted as 조사(JX) or 보통명사(NNG). The second one is because of the characteristic of an agglutinative language which results in many derivatives of single verb. PoS tagging can handle this problem pretty well by picking out the root word of derivatives.

| Before PoS Tagging | After Pos Tagging |
|---|---|
| 걷다가 | '걷/VV', '다가/ECD' |
| 걸어서 | '걷/VV', '어서/ECD' |
| 걸으니 | '걷/VV', '으니/ECD' |

*Table 2.* Splitting based on the root word

### 5.2. Minimizing the number of named entities

The novel usually contains many characters, more than 10 is not a rare case even in short stories. However, named entities can dominate the sentence vector. They should be minimized as much as they could to avoid negative influence in training the models.

### 5.3. Sentence tagging

Sentence tagging was inspired by [1]. According to the author, the scene - a basic unit of the novel - works as one of four chords of fiction [1]. The two major chords are action and reaction, while the two minor chords are setup and deepening. Action and reaction tend to dominate, with the minor chords dropping in. He also claims that concatenating scenes can simply finish writing a novel.

What we have done is tuning his idea a little bit so the model can utilize it. We set our goal to generate a scene by stacking three types of sentences instead of building a novel with four chords of scenes. We sorted sentences out with 3 main tags - major sentence(a), dialogue(b) and minor sentence(c) - and divided the second part, the dialogue, into beginning(x), middle(y) and end(z). In addition to that, since there are many interrogative sentences in the dialogue, another tag(q) was added to mark on.

| Tag | Meaning |
|---|---|
| a | Major sentence |
| c | Minor sentence |
| b x | Dialogue, Beginning |
| b y | Dialogue, Middle |
| b z | Dialogue, End |
| b x/y/z q | Dialogue, Interrogative sentence |

*Table 3.* All cases of sentence tagging

To explain each group in detail, major sentence refers to a sentence that is crucially related to the main plot. If this sentence is removed from the novel, the story becomes really awkward and nonsensical. minor sentence refers to a sentence that is opposite to the major sentence. Even though this sentence is not contained in the novel, it has no problem for readers to understand the content. But they are important because they work like a seasoning in the food which keeps the reader from being tired of following the main plot. In terms of Doc2Vec, each document vector is comprised of the vectorized set of PoS tagged words in a given sentence, including a sentence tag - that we have additionally put - that can have an effect on making the document vector. Running with this processed data, the model is able to understand how the sentences are proceeded. Below is the example of sentence tagging.

Before sentence tagging: 시끌시끌하던 예진이의 방쪽으로 슬쩍 고개를 돌려봤던 준호는 폭 한숨을 내쉬며 다시 영단어집으로 시선을 끌어내렸다. 하지만 저쪽방이 하도 소란스러워서 이내 책을 덮어버렸다. “아우 진짜 저게 누나만 아니었으면 확 그냥” “아니었음 어쩌려고 우리가 질게 뻔하잖아?” “하긴” 이 집의 맏딸이자 이 쌍둥이의 누나인 김전영 양은 키도 160이 채 안되고 전체적으로 몸이 작지만 특공무술 유단자이다.

After sentence tagging: 시끌시끌하던 예진이의 방쪽으로 슬쩍 고개를 돌려봤던 준호는 폭 한숨을 내쉬며 다시 영단어집으로 시선을 끌어내렸다 a. 하지만 저쪽방이 하도 소란스러워서 이내 책을 덮어버렸다 a. 아우 진짜 저게 누나만 아니었으면 확 그냥 b x. 아니었음 어쩌려고 우리가 질게 뻔하잖아 b y q. 하긴 b z. 이 집의 맏딸이자 이 쌍둥이의 누나인 김전영 양은 키도 160이 채 안되고 전체적으로 몸이 작지만 특공무술 유단자이다 c.

## 6. Models

### 6.1. Temperature function

Temperature function is mostly used to normalize values within the softmax function. As avoid overfitting was essential in predicting the next sentence, we implemented it inside the model and optimized the parameter empirically. The application of it can be summarized as the following order.

1. The BiLSTM model predicted a vector A.
2. Found top N vectors that are similar to the predicted vector using cosine similarity.
3. Normalized the cosine similarities to consider them as probabilities.
4. Adjusted the calculated probabilities using the temperature function. The gap between probabilities can be shrank or expanded.
5. Picked one vector randomly with the given probabilities.

### 6.2. Human based greedy search

The term human based greedy search has already been used by a former researcher, A. Rakotomamonjy, in his work of acoustic scene classification with CNN architecture [18]. The main difference to usual greedy search is the human involvement in selecting the best one. We made 5 slightly different models using BiLSTM and Doc2Vec that recommends the most suitable next sentence from the previous sentence or sentences. When each of them spits out a recommending sentence, the human is forced to choose among 5 of them. This process is repeated for pre-set times.

#### 6.2.1. MODEL 1

The first model is not different from the simple feed-forward deep learning architecture. It only uses the previous sentence to predict the next one. The generating process starts off by embedding each sentence in the dataset $S$ to $X$ using $g_1$,

$$S = \{s_1, s_2, \cdots, s_k\} \tag{21}$$

$$X = \{x_1, x_2, \cdots, x_k\} \tag{22}$$

where $g_1 : S \to X$ is the embedding function and $k$ is the total number of sentences in the training dataset. And then, we calculate the $n$-dimensional vector $\hat{x}_{t+1}$ with given input $x_t$, where $n$ is the dimension of a document vector.

$$\hat{x}_{t+1} = f_1(x_t). \tag{23}$$

We consider the vector generated by $f_1(x)$ as the output by the network shared through model 1 to model 3. Likewise $f_2(x)$ - used in model 4 - is the output by the second network and the last one $f_3(x)$ - used in model 5 - is the output by the third network.

Next we find the total of top 7 similar vectors to the predicted one with their ordered statistics and regard them as the candidates of next sentence by getting cosine similarities with each of the sentences in the dataset. The detailed calculation is given by,

$$\hat{x}_{t+1,i} = \frac{\hat{x}_{t+1} \cdot x_i}{||\hat{x}_{t+1}|| \cdot ||x_i||},\ i = 1, ..., k \tag{24}$$

$$X_k = max\{\hat{x}_{t+1,1}, \hat{x}_{t+1,2}, \cdots, \hat{x}_{t+1,k}\}. \tag{25}$$

Among the candidates, one is picked as $x_{t+1}$, with its probability and the same process is repeated with $x_{t+1}$.

$$\text{candidates} = \{X_k,\ X_{k-1},\ X_{k-2},\ X_{k-3},\ X_{k-4},\ X_{k-5},\ X_{k-6}\}.$$

With respect to the novel, $s_{t+1}$ is concatenated as the next sentence.

#### 6.2.2. MODEL 2

The second model uses two previous sentences to predict the next one. Likewise the total of 7 were picked and then one of them is finally picked with probabilities. Since the very first prediction cannot have the last two previous sentences, the first model is used at the first prediction.

$$\hat{x}_{t+1} = f_1(x_{t-1}, x_t). \tag{26}$$

#### 6.2.3. MODEL 3

The third model uses all previous sentences to predict the next one. The other conditions are the same as the second model. The 3 models mentioned share the same document vectors for both training and prediction.

$$\hat{x}_{t+1} = f_1(x_1, \ldots, x_{t-1}, x_t). \tag{27}$$

#### 6.2.4. MODEL 4

The fourth and fifth model has a distinct architecture while the three above share the same document vectors and model structure.

$$\hat{x}_{t+1} = f_2(x_t). \tag{28}$$

$$x_{t+1} = g_2(s_{t+1}{}^\frown s_{t+2}),\ x_t = g_2(s_{t-1}{}^\frown s_t)$$

$$\text{where } g_2 : S \to X.$$

The notation $^\frown$ is used to mean concatenation. In this model two sentences are concatenated and regarded as a basic unit. Using this, another document vector set was made for training and prediction. After getting $x_{t+1}$, we only use $s_{t+1}$ and concatenate it with $s_t$ while $s_{t+2}$ is being discarded. In conclusion, the next input $x_{t+1}$ is the same as $s_t{}^\frown s_{t+1}$ and the prediction is processed further with it.

#### 6.2.5. MODEL 5

The only difference between the fourth and fifth model is that the latter regarded the 3 concatenated sentences as a basic unit instead of two.

$$\hat{x}_{t+1} = f_3(x_t). \tag{29}$$

$$x_{t+1} = g_2(s_{t+1}{}^\frown s_{t+2}{}^\frown x_{t+3}),\ x_t = g_2(s_{t-2}{}^\frown s_{t-1}{}^\frown s_t)$$

$$\text{where } g_3 : S \to X.$$

## 7. Experiments

### 7.1. Generating procedure

The whole process of the experiment can be summed up as below.

1. Went through sentence embedding.
2. Trained 5 models with embedded sentences.
3. Predicted next sentence from each of the models. The first sentence was picked randomly from the dataset.
4. Chose one among 5 recommended sentences. A scene was made by repeating this for user-given times.
5. A novel is finished by concatenating scenes.

### 7.2. Model architecture

Two BiLSTM layers were stacked, and the pooling layer was added on top of them. Parameters were chosen after many trials and tests but it was tough because the evaluation index to appreciate good novels was still absent. Mean squared error was used as a loss function instead of cross-entropy loss function since it was more suitable to capture the error between real vector and the predicted one. Lastly, adam optimizer was used.

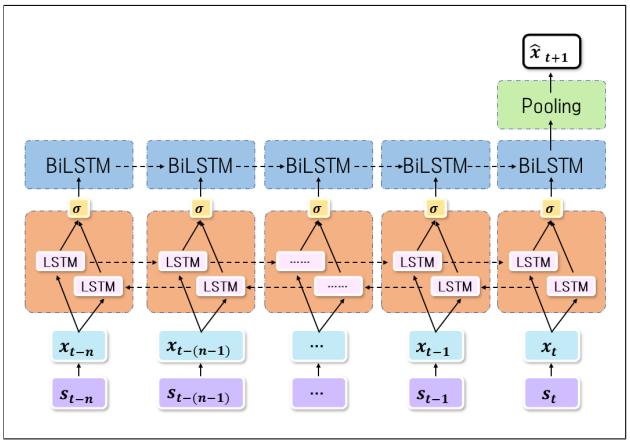


*Figure 5.* The total architecture of model 3

### 7.3. Final output


*Figure 6.* Cover photo of *BanHangA*

The title of the novel, *BanHangA*, is named after one of the training dataset's title meaning rebellious child. There are three main characters - 김준호(Kim, Jun-Ho), 박서연(Park, Seo-Yeon) and 김예진(Kim, Ye-Jin) - in the story and the genre is romance. Speaking of the story briefly,

*Jun-Ho is a model student that everybody likes. He gets good grades in school, and is participating in a high school band as an extracurricular activity to become a singer. Ye-Jin is his best friend. She loves him but doesn't have enough courage to get closer because of the fear of being estranged from him since they have been just friends for a long time. Seo-Yeon is Jun-Ho's ex-girlfriend who still has an affection for him. While Jun-Ho is trying to ignore her behaviors intended to draw his attention because the relationship with her has made his grade down even though he still has feelings for her as well.*

One of the scenes that the model had made is the following below.

여태 강남에서 태어나서 강남에서 아파트만 보고 살았던 준호는 드라마에나 나올 것 같은 이 좁고 집들이 다닥다닥한 오르막 골목에 적응이 안된다. 가족끼리 나와서 배드민턴을 치는 사람들도 좀 있고 삐에로 옷을 입은 사람이 길다란 풍선으로 푸들이나 인형같은 걸 만들어서 들고 다니며 팔기도 하는 마로니에 공원은 꽤 북적이는 듯하다. 아무래도 추울 것 같다고 하면서 제 점퍼를 벗은 예진이는 준호의 어깨에 제 옷을 둘러줬다. 100분 동안의 라이브 공연 연극을 보고 나온 두 사람은 각자 간단한 연극의 품평을 하기도 했지만 제발 밥을 좀 먹었으면 좋겠다는 말을 더 많이 했다. 어차피 밤 10시가 다 되어가니 이제 정말 다들 집에 가자는 말을 하면서 일어난 예진이를 선두로 분식집에서 나온 두 사람은 여기 왔을때와 마찬가지로 마로니에 공원 앞을 지나가게 되었다. 도저히 일행이라고 하기가 쪽팔려서 남은 돈을 차비빼고 다 털어서 거국적으로 김밥집을 습격했던 두 사람이 냅킨으로 입을 닦으며 잘먹었다는 말을 중얼거리는데 예진이의 엄마가 이 녀석 왜 이렇게 안오는 것이냐며 전화를 하셨다. 준호는 혼자 조용히 앉아서 바닥으로 시선을 잔뜩 내려놓은 채 낮게 흥얼거린다.

"몇시까지 가?"

"이따가 오면 전화해"

"어딜 도망가냐"

준호에게 예진이는 왜 그러냐고 물었다. 준호는 예진이의 입을 확 틀어막아버렸다. 그리고 박서연의 그늘에 가려서 그닥 빛을 못 보긴 했지만 그래도 골수팬이나 마니아는 꽤 갖고 있던 김예진. 벌게진 서로의 얼굴만 잔뜩 쳐다보던 두명은 살며시 한숨을 내쉬고 조용히 길을 걸었다. 예진이는 그 와중에도 준호에게 그러게 너도 나랑 같이 디지털 사진부 가입했으면 좋았잖아라고 칭얼거렸다.

The full novel with 3 episodes is submitted to an mo-

bile application Blice that is made by a South Korean telecommunication company Korea Telecom, and is introduced to the public online as one of the novels written by artificial intelligence.

## 8. Conclusion and Future Work

The main drawback of A.I. written novels is that it does not have a big flow but a group of small streams. This is often a trend found in the latest web-novels but hard to be called as a flawless novel. Nevertheless, there are novels without plots. For example, the novelist D. Richardson wrote 'Pilgrimage' on 1915 using one of the famous writing technique nowadays, the stream of consciousness which proved that a novel can have a deviate format [10]. It was criticized not following conventional writing style of that period but the assessment of her is quite opposite these days. Likewise, a new genre of art can be created from somewhere that was not expected at the beginning. Last but not least, the index to evaluate good novels is remained as a future work. This would make computers to generate a novel with pure beauty.

## References

[1] Bell, J. S., Write great fiction: plot and structure, Writer's Digest Books, pp.113-129 (2004)

[2] Bengio, Y. and Ducharme, R., Vincent, P. and Janvin, C.. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137–1155 (2003)

[3] Bengio, Y., Frasconi, P., and Simard, P. The problem of learning long-term dependencies in recurrent networks. IEEE-ICNN93, San Francisco, CA, pp.1183-1188 (1993)

[4] Chung, J., Gulcehre, C., Cho, K. and Bengio Y., Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)

[5] Cui, Z., Ke, R. and Wang, Y., Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic Speed Prediction. arXiv preprint arXiv:1801.02143 (2018)

[6] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R, and Schmidhuber, J. LSTM: A Search Space Odyssey. arXiv preprint arXiv:1503.04069 (2015)

[7] Graves, A. and Schmidhuber, J., Framewise phoneme classification with bidirectional LSTM networks, IEEE International Joint Conference on Neural Networks (IJCNN) (2005)

[8] Hochreiter, S. and Schmidhuber, J., Long short term memory, Neural Computation, Vol.9(8), pp.1735–1780 (1997)

[9] Huang, Z., Bidirectional LSTM-CRF models for sequence tagging, arXiv preprint arXiv:1508.01991 (2015)

[10] Kim, B., A Novel without a plot: Modernist readership and boredom in Dorothy Richardson's Pilgrimage, Feminist Studies in English Literature Vol.25(2) (2017)

[11] Le, Q. and Mikolov, T., Distributed representations of sentences and documents. International Conference on Machine Learning (2014)

[12] Lee, J., A study on creative writing by artificial intelligence and the destiny of the author, The Journal of Korean Fiction Research (2017)

[13] Liu, B., Fu, J., Kato, M. P., Yoshikawa, M., Beyond narrative description: generating poetry from images by multi-adversarial training, arXiv preprint arXiv:1804.08473 (2018)

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

[15] Olah, C., Understanding LSTM networks. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (2015)

[16] Park, E. L. and Cho, S., KoNLPy: Korean natural language processing in Python (2014)

[17] Pascanu, R., Mikolov, T. and Bengio, Y., On the difficulty of training recurrent neural networks. arXiv preprint arXiv:1211.5063 (2012)

[18] Rakotomamonjy, A., DCASE 2017 - Task 1 : Human-based greedy search of CNN architecture, detection and classification of acoustic scenes and events (2017)

[19] Rong, X., word2vec Parameter Learning Explained, arXiv preprint arXiv:1411.2738 (2014)

[20] Schuster, M. and Paliwal, K. K., Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing, Vol.45(11), pp.2673–2681 (1997)

[21] Yi, X., Li, R. and Sun, M., Generating chinese classical poems with RNN encoder-decoder. arXiv preprint arXiv:1604.01537 (2016)

[22] Zheng, C., Zhai, S. and Zhang, Z., A Deep Learning Approach for Expert Identification in Question Answering Communities, arXiv preprint arXiv:1711.05350 (2017)