Deep Lyrics Generation: Pop Song Transformation into K-pop While Keeping the Rhyme

Korea University COSE461 Final Project

Suwon Son
Department of Computer Science
Team 24
2019320135

Dain Kim
Department of Computer Science
Team 24
2019320093

Abstract

The paper describes the implementation and evaluation of a system able to generate Korean lyrics while keeping the rhyme of the input pop song. The lyrics generation system first extracts the last word of each line of the input English lyrics and finds the corresponding Korean word with the most similar pronunciation using nearest neighbor algorithm. The system then generates Korean lyrics whose lines end with the corresponding words found earlier using a Long Short-Term Memory (LSTM) Neural Network trained on K-pop song lyrics dataset. Billboard's annual number one songs were transformed into K-pop indicating that the deep learning based system is capable of generating Korean lyrics.

1 Introduction

Nowadays, in K-pop market, it's common to buy a song from abroad whose lyrics is English and transform it into Korean. While doing this, enterprise and lyricist pay attention to keeping the rhyme of the original lyrics in order to bring out the composer's intention and musicality as much as possible. The goal of this project is to implement deep learning based system for this process.

Sung-Hwan et al. trained data that is reversed by the unit of a letter, however this method generated grammatically incorrect phrases. In order to solve the problem, we reversed each sentence by word unit instead of letter. In our model, reversed sequences are given as an input and ouput is calculated by LSTM Cell using the embedding vector and hidden units. The weights of embedding and LSTM layer are updated by cross entropy loss.

We collected K-pop lyrics of the top 100 songs on the annual chart for 40 years from 1980 to 2019 as training dataset. In order to transfrom a pop song into K-pop, we had to find Korean words with similar pronunciation with corresponding English words located at the end of each line of original lyrics. We created rhyme pool dictionary that matches romanized Korean word with its CMU based pronunciation vector embedding. Test dataset was made using Billboard's annual number on songs. End words of a song were extracted and rhyme pairs were found using nearest neighborhood algorithm to measure vector embedding similarity. These rhyme pairs are given as an input of trained LSTM model.

2 Related Work

Lyrics creation and songwriting have a lot in common. In fact, the field of poetry generation often use song lyrics rather than poetry as their dataset since collecting a dataset consisting of poetry is very inconsistent, meaning that every poet writes differently and uses different words and expressions.

Furthermore, song lyrics dataset leads low perplexity since network tries to mimic the property of song lyrics: using small variety of words and repeated lines. (Malte and Bjorn 2018) To generate rap lyrics, Eric et al. (2016) complied a list of 104 popular English-speaking rap artist and scraped all their songs available on a popular lyrics website. Sung-Hwan et al. (2019) used a set of ballad lyrics collected from a music site called 'MELON' to generate Korean lyrics. Malte and Bjorn (2018) collected their dataset from www.mldb.org, an online song lyrics database for poetry generation.

To generate poetry with a fixed form, Malte and Bjorn (2018) had to find words with similar rhymes to be located at the end of each row. This was done using CMUdict, an open-source machine-readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations. Two rhyming rules were used to find the rhyming words. The first checked if the endings of the words were pronounced the same. The second rule stated that the consonant sounds preceding the rhyme must be different so that slant rhymes cannot be allowed in the system. Malte and Bjorn called these words with similar rhymes 'rhyme pairs'.

Malte and Bjorn (2018) used LSTM network with four layers to generate poetry: one input layer to represent the words in the dataset, two hidden layers with the size of 1100 cells to compute the possible values for the next predictions, and one softmax layer. Sung-Hwan et al. (2019) aimed to create Korean lyrics. Therefore, they focused on generating lyrics based on the predicate reflecting the factor that Korean is cohesive language. Since Korean has its predicate in the latter part of the sentence, they reversed sentences so that the predicate is located at the beginning of each sentence where model starts learning from. They used character unit Korean language model with LSTM predicting next character based on the input text.

Sung-Hwan et al. (2019) performed a human evaluation on the generated song lyrics to check whether generated context in the model is a natural flow like a normal human-spoken language. The human evaluation is calculated by dividing the number of bars that are grammatically correct and natural in context by total number of generated bars. Malte and Bjorn (2018) also did human-based evaluation. They evaluated output in three aspects: grammaticality, meaningfulness, poeticness. Each of the dimensions was scored on a scale of 1-3.

3 Approach

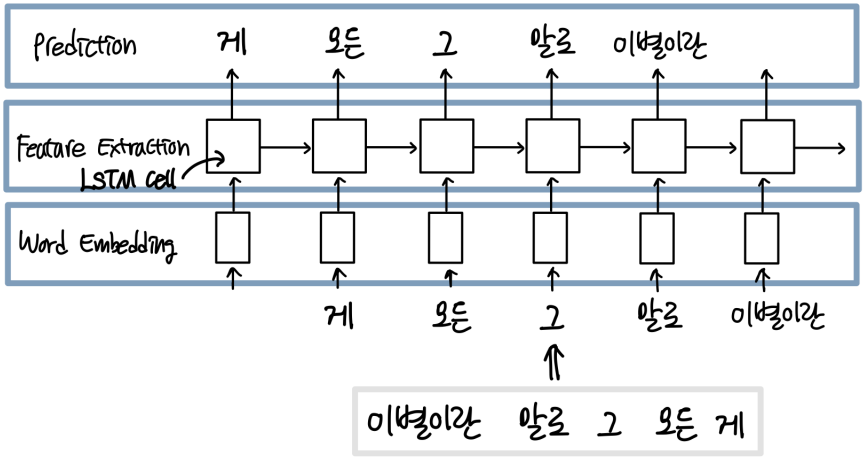
3.1 Baseline

To generate a sentence starting with the predicate and connection words that determine the context of the lyrics, Sung-Hwan Son et al.(2018) trained data that is reversed by the unit of a letter. Lyrics correlated each other in context were generated, but there was a problem of grammatically incorrect phrases or sentences.

3.2 Model

In order to solve the problem of the previous study, we reversed each sentences by word unit rather than letter unit.

'조금씩 잊혀져 간다' -> '다간 져혀잊 씩금조' (Baseline)
'조금씩 잊혀져 간다' -> '간다 잊혀져 조금씩' (Reverse by Word Unit)



Embedding vectors of sequences are generated by entering reversed sequences of input into the embedding layer. Output calculated by LSTM Cell using the embedding vector and hidden units is an embedding vector of a word being predicted for next location. Output from the sequence is compared to label and cross entropy loss is calculated. Then weights of embedding layer and LSTM layer are updated by cross entropy loss.

$$\hat{P}(x_t|context) \approx P\big(x_t|LSTM(x_0, x_1, \ldots, x_{t-1})\big)$$

When generating sentences, the language model selects a word with highest probability of being located next for a given sequence. It generates sentences in reverse order as it trained data in reverse.

3.3 Preprocess Data

However, in word-by-word training, a word set consisting of more than 30,000 words was formed, making it impossible to encode labels of each sequence into one-hot vectors. For smooth training, it was necessary to reduce the size of the word set, and we approached the problem in three ways: 1) tokenize words using KoNLPy toolkit, 2) remove rare words that are less frequent from the word set, 3) split data into small size.

- Tokenize Words using KoNLPy
  Though using KoNLPy, there were many words that were not properly separated into word root, postposition and connection words. As there was no significant change from not using KoNLPy, we also removed the rare words with using KoNLPy.

- Remove Rare Words from Word Set
  In order to generate a word set with about 3500 words of high frequency, words with relatively low frequency were defined as rare words. The rare words that classified as OOV and excluded from the word set were about 70%-80% of total data. Proper training was difficult due to the large proportion of the rare words. Also, as starting words in generating step did not exist in the word set, the model generated sentences only using the highly frequent words in the entire data without considering the context.

  이 나 는 나 는 위에서
  이 나 는 나 는 아닌건지
  이 나 는 나 는 그대여
  넌 내 가 너 를 찾아가
  너 너 너 너 너 너

- Split Data

To reduce the proportion of rare words and proceed training smoothly, data was divided into the unit of 5 years. Training with each unit of data, 8 training sessions were conducted.

4 Experiments

4.1 Data

As Sung-Hwan et al. (2019) did, we collected K-pop lyrics from a music site called 'MELON'. By crawling lyrics of the top 100 songs on the annual chart for 40 years from 1980 to 2019, we could collect 3719 songs. (100 songs were extracted every year from 1985 to 2019. The site only had the data of 219 songs from 1980 to 1984.) It was advantageous to compose dataset using MELON since it has been one of the most popular music sites in Korea where most people listen to K-pop instead of pop songs. It is hard for pop songs to be in Korean music charts so we could get K-pop lyrics data from this site by simply crawling on the annual chart in order of popularity. The extracted lyrics were sliced into lines(sentences) based on newline character. The dataset was filtered to remove lines that included special characters and non-Korean characters. The spacing at the beginning and end of the sentences were also deleted.

In this study, in order to transform a pop song into K-pop, we had to find Korean words with similar pronunciation to those located at the end of each line of English lyrics subject to transform to form a rhyme pair. Korean Romanizer open source, CMU Dictionary, and nearest neighborhood algorithm were used to compare the pronunciation of Korean and English words. From previously created dataset, we first collected end words of each sentence. Only the end words were collected since by doing so, the model would learn which words are appropriate to end the sentence. Using these words and Korean Romanizer, dictionary matching Korean words and their romanized version as they sound was created. Let us call this romanized version 'romanization'.

Allison (2017)'s algorithm was used to generate and compare pronunciation vector embedding of the words. Since we are dealing with romanization which would not be in dictionary, LOGIOS Lexicon Tool was needed. LOGIOS Lexicon Tool is used to express the pronunciation of words that are not in the dictionary. It basically uses CMU dictionary along with some simple normalization and inflection rules. Using this tool, we could obtain dictionary matching romanization and its CMU machine-readable pronunciation. By putting this CMU based pronunciation dictionary into Allison's algorithm, we could create another dictionary matching romanization and its pronunciation vector embedding. Let us call this dictionary 'rhyme pool'.

Test dataset was made in similar way. Billboard's annual number one songs were used. End words of a song were extracted. By using LOGIOS Lexicon Tool and Allison's algorithm, we could obtain dictionary matching end word itself and its CMU based pronunciation vector embedding. Annoy library was used to choose a Korean word from the rhyme pool whose pronunciation is the most similar to each word of test lyrics. Annoy library uses nearest neighborhood algorithm to measure vector similarity. For each song, text file containing rhyme pairs corresponding to each line is generated, and this is given as an input of trained model.

4.2 Evaluation method

In the field of song lyrics and poetry generation, human evaluation is often used as an evaluation metric. Malte and Bjorn (2018) evaluated their output in three aspects: grammaticality, meaningfulness, poeticness. Similarly, in this project, output will be evaluated in three aspects: grammaticality (written according to Korean grammar however accepting poetic license to a certain extent considering the characteristics of song lyrics), meaningfulness(convey a conceptual message which is meaningful under some interpretation), and rhyming (keeping the rhyme of original lyrics).

4.3 Experimental details

In order to prevent overfitting, the size of the embedding vector is not set larger than 100 and the number of LSTM layer and hidden units are set small. It took about 3 hours to train the entire data using the word set consisting of about 3,500 words. When the data was divided, though the size of the word set became larger, it took about 30 minutes per training because of the small data.

Embedding Dimension	100
LSTM Layers	1
Hidden Units	32
Dropout	0.2

Batch Size	32
Epoch	100
Learning Rate	0.001

4.4 Results

When the rare words were input, word arrangements that frequently appear in the training data appeared repeatedly. Overfitting was also found to use the lyrics of training data as they are.

Released Year of Training Data	Generated Lyrics
1980-1984	나의 내사랑 바람에 흩어진 이 놓나 나의 내사랑 바람에 흩어진 이 지펴줘 나의 내사랑 바람에 흩어진 이 게 나의 내사랑 바람에 흩어진 이 라라라라라
1990-1994	나의 처음에 만난 그 놓나 나의 처음에 만난 그 지펴줘 나의 처음에 만난 그 주었던 나의 처음에 만난 그 생각나죠
2000-2004	수 없는 너의 그 놓나 수 없는 너의 그 지펴줘 수 없는 너의 그 널렸어 수 없는 너의 그 줘

Generating Repeated Words

나 다시 태어나도 너만을 사랑할 거야
오늘 이별이 힘들었단 걸 잊지마
내가 사랑한단 그런말을 왜 하나요
하지만 이제는 먼 그리움 되어
성당의 종소리만이 희미하게 너의눈물을 위로해
저 하늘이 외면하는 그 순간
...

Generation Overfitting

When the words present in the word set were input, they produced much more natural lyrics than the baseline. There was no grammatically incorrect phrase because the words used in the training data were used without modification. However, because of training a small amount of data, grammatically incorrect examples appeared when looking at the entire sentence.

1985-1989	왜이리 내 마음 흔들어 놓나 다시 있지만 내겐 하나뿐인 너 꿈같이 흘러간 사람은 언제나 그댄
1990-1994	이렇게 우린 스쳐가는 지난 세월속에서 다시 나의 사랑을 찾아 기다리고 또 아무말도 서글퍼 바쁘기만한 시계

In order to find similar pronunciation words from the descriptive and connective words that determine the context of the sentence, we tried to search phoneme only from words located at the end of each sentences. However, since the size of the word set of descriptive words and conjunctions was not large, there was a limit to finding words with similar pronunciation to English lyrics.

Wake up in the mornin' feelin' like P Diddy
왜이리 내 마음 흔들어 놓나

Before I leave, brush my teeth with a bottle of Jack
다시 있지만 내겐 하나뿐인 너

5 Analysis

The preprocessing work of separating the root from words should have been done in more detail. Also, when a rare word was input, only highly frequent words were repeatedly generated. This is a problem that occurs when generating a sentence only using the word with the highest probability of being located next. This problem will be improved if random sampling is performed on a group of predicate words above a certain probability for being located next. To improve the pronunciation similarity to English lyrics, we should search words from the entire set of words.

6 Conclusion

Among Korean words with similar pronunciation to English lyrics, we tried to create sentences concentrating on the descriptive and connective words which represent the context of the lyrics. Unlike previous studies, which conducted training by reversing sentences in letter units, sentences were reversed in word units to generate sentences starting with descriptive words. The goal of this experiment was to create Korean lyrics with similar pronunciation to English lyrics, but as the size of the word set was small, similar words could not be found. In addition, there was a limitation that the amount of data for training should be set small due to the lack of GPU capacity. Due to the limitation, the size of the word set required for sentence generation and the number of learning data were adjusted to be small, resulting in problems of overfitting and meaningless word repetition. However in the process of analyzing the cause of this problem, we learned what we should do to generate various sentences: reduce the size of the word set through more detailed word root analysis and select next words with random sampling.

References

[1] Malte Loller-Andersen and Björn Gambäck. Deep learning-based poetry generation given visual input. In *ICCC*, pages 240–247, 2018.

[2] Sung-Hwan Son, Hyun-Young Lee, Gyu-Hyeon Nam, and Seung-Shik Kang. Korean song-lyrics generation by deep learning. In *Proceedings of the 2019 4th International Conference on Intelligent Information Technology*, pages 96–100, 2019.

[3] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 195–204, 2016.

[4] Allison Parrish. Poetic sound similarity vectors using phonetic features. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.

[1] [2] [3] [4]

A Appendix: Team contributions

The stuff here does not count towards the 8 page limit. If you are a multi-person team, you can write a brief summary of what each team member did for the project (about 1 or 2 sentences per person). For almost all teams, it will have no effect (i.e. team members all receive same grade), but for teams with considerably unequal contribution, I may investigate and/or give different grades to team members.

Suwon Son

- Crawling lyrics from Melon website & Preprocess data for training
- Construct model & Model training

Dain Kim

- Topic suggestion
- Construct training and test dataset & obtain vector embeddings and rhyme pairs