석사학위논문

딥러닝 기반 자동작곡에서 마디 임베딩을 이용한 곡의 생성 및 정량적 평가 방법에 대한 연구

A Study on the Method of Creating and Quantitatively Evaluating Songs using bar Embedding in Automatic Composition Based on Deep Learning

2021년

한성대학교 지식서비스&컨설팅대학원

미래융합컨설팅학과

미래융합전략컨설팅전공

이 영 배

석 사 학 위 논 문
지도교수 정성훈

딥러닝 기반 자동작곡에서 마디 임베딩을 이용한 곡의 생성 및 정량적 평가 방법에 대한 연구

A Study on the Method of Creating and Quantitatively Evaluating Songs using bar Embedding in Automatic Composition Based on Deep Learning

2020년 12월 일

한성대학교 지식서비스&컨설팅대학원

미래융합컨설팅학과

미래융합전략컨설팅전공

이 영 배

석 사 학 위 논 문
지도교수 정성훈

딥러닝 기반 자동작곡에서 마디 임베딩을 이용한 곡의 생성 및 정량적 평가 방법에 대한 연구

A Study on the Method of Creating and Quantitatively Evaluating Songs using bar Embedding in Automatic Composition Based on Deep Learning

위 논문을 컨설팅학 석사학위 논문으로 제출함

2020년 12월 일

한성대학교 지식서비스&컨설팅대학원

미래융합컨설팅학과

미래융합전략컨설팅전공

이 영 배

이영배의 컨설팅학 석사학위논문을 인준함

2020년 12월 일

심사위원장 _____(인)

심 사 위 원 _____(인)

심 사 위 원 _____(인)

국 문 초 록

딥러닝 기반 자동 작곡에서 마디 임베딩을 이용한 곡의 생성 및 정량적 평가 방법에 대한 연구

한성대학교 지식서비스&컨설팅대학원
미 래 융 합 컨 설 팅 학 과
미 래 융 합 전 략 컨 설 팅 전 공
이 영 배

딥러닝(Deep Learning)을 이용하여 음악을 생성하는 연구는 다양하게 진행이 되어 왔으나, 생성된 음악에 대한 평가 방법 연구는 상대적으로 적다. 생성된 곡(song)의 품질을 평가할 수 없는 상태에서 자동 작곡 모델의 성능을 개선하기는 어렵기 때문에 평가는 매우 중요하며, 평가에는 또한 기준이 중요하다. 아직까지 가장 완성도가 높은 음악은 인간이 만든 곡이기 때문에 우리는 인간이 작곡한 곡을 평가의 기준으로 설정해서 자동 작곡 모델이 인간이 작곡한 곡과 얼마나 유사하게 곡을 생성하는지에 대한 정량적 평가를 진행하였으며, 인간이 작곡한 곡과 유사하게 곡을 생성할수록 평가 점수가 높게 나올 수 있는 평가 지표를 가진 정량적 평가 방법이 필요하였다. 이전 연구자들이 연구한 정량적인 평가 방법이 풍부하지 않기 때문에 본 연구에서는 기계 번역(Machine Translation, MT)이나 이미지 캡셔닝(Image Captioning) 같은 딥러닝의 다른 분야에서 사용되는 정량적 평가 방법들에 대해서 자세하게 분석한 후, 이를 딥러닝 기반 자동 작곡 모델 평가에 적용해보았다.

모든 자동 작곡 모델은 수학적인 모델이기 때문에 모델의 구현을 위해서는 음표를 모델이 인식할 수 있는 수치 데이터(numerical data)로 변환해야 한다. 음표를 수치 데이터로 변환하는 방식에는 0과 1의 2가지 정수만을 이용하는 원-핫 인코딩(one-hot encoding) 방식과 자연어 치리 분야에서 단어를 변환하는 방식인 단어 임베딩을 응용해서 구현한 음표 임베딩(note embedding) 방식이 있는데 본 연구에서는 두 가지 방식을 모두 사용하여 자동 작곡 모델을 구현하였으며 앞에서 서술한 바와 같이 딥러닝의 다른 분야에서 사용되는 정량적 평가 방법을 이용하여 이 두 가지 방식 중에서 어느 방식이 인간이 작곡한 곡과 더 유사하게 곡을 생성하는지를 평가하였다. 뿐만 아니라 자연어 처리 분야에서 문자 임베딩과 단어 임베딩이 가능하듯이 음악에는 음표와 마디가 있으므로 임베딩을 적용하는 단위를 세분화하여 새롭게 마디 임베딩(bar embedding) 방식을 구현하였으며 기존의 음표 임베딩 방식과 비교하여 어느 방식이 인간이 작곡한 곡과 더 유사하게 곡을 생성하는지를 정량적으로 평가하였다.

이를 통해서 딥러닝의 다른 분야의 정량적 평가 방법을 자동 작곡 분야에 적용함에 있어서 적용이 가능한 것과 적용할 수 없는 것을 확인하였으며, 원-핫 인코딩 방식과 음표 임베딩 방식 중에서는 음표 임베딩 방식의 평가 점수가 더 높았으며, 이번에 새롭게 구현한 마디 임베딩 방식과 음표 임베딩 방식 중에서는 마디 임베딩 방식의 평가 점수가 더 높았다. 이번 연구 결과를 통해서 기존의 원-핫 인코딩 방식이나 음표 임베딩 방식보다 마디 임베딩 방식이 인간이 작곡한 곡과 더 유사하게 곡을 생성한다는 사실을 확인하였기에 자동 작곡에서 마디 임베딩 방식을 사용하기를 제안한다.

【주요어】 자동작곡, 정량적 평가, 원-핫 인코딩, 음표 임베딩, 마디 임베딩

목 차

표 목 차

그 림 목 차

Ⅰ. 서론

1.1 연구 배경

음악을 평가하는 방식에는 정성적인 방식과 정량적인 방식이 존재한다. 정성적 평가 방법은 사람이 직접 생성된 곡을 듣고 평가하는 것이고, 정량적 평가 방법은 수학적 연산을 통해서 평가하는 것이다[Yang. L. C. 등, 2018]. 음악은 원래 인간이 듣기 좋으라고 만든 것이므로, 정성적 평가 방법이 더 합리적이고 합당한 평가 방법이라고 생각할 수 있다. 그러나 정성적 평가의 경우 음악이 가지는 창의성이나 아름다움, 예술적 가치를 평가할 수 있는 평가 지표의 개발, 모든 관련 변수를 제어하고 편향을 제거하며 충분한 수의 자격 있는 피험자들을 모집하는 것 같은 평가를 위한 엄밀한 설계를 하는 것이 매우 어렵고, 평가를 수행하기 위한 비용도 많이 든다[Logan. B. 등, 2003 : Yang. L. C. 등, 2018].

따라서 정성적인 평가를 대체할 정량적인 평가 방법이 필요하다. 아직까지는 이전 연구자들이 연구한 정량적인 평가 방법이 풍부하지 않기 때문에 기계 번역이나 이미지 캡셔닝과 같은 딥러닝의 다른 분야에서 사용되는 정량적인 평가 방법들이 자동 작곡 분야에도 적용이 가능한지에 대한 연구가 필요하다. 기계 번역과 같은 분야에서 사용되는 정량적 평가 방법들은 정확도(Precision)나 재현율(Recall)과 같은 평가 지표들을 이용하여 번역 모델이 생성한 문장이 인간이 번역한 문장과 유사할수록 번역 모델의 성능이 우수하다고 평가한다[Papineni. K. 등, 2002 : Lin, C. Y. 2004 : Banerjee, S. 등, 2005]. 우리는 인간이 작곡한 곡을 평가의 기준으로 설정해서 자동 작곡 모델이 인간이 작곡한 곡과 얼마나 유사하게 곡을 생성하는지에 대한 정량적 평가를 진행하고자 한다. 이에 기계 번역과 같은 분야에서 사용되는 정량적 평가 방법들이 우리의 목적에 적합한지에 대한 조사와 연구가 필요하다.

1.2 연구 내용

본 연구의 목적은 인간이 작곡한 곡을 평가의 기준으로 설정해서 자동 작곡 모델이 인간이 작곡한 곡과 얼마나 유사하게 곡을 생성하는지에 대한 정량적 평가를 진행하는 것이다. 이와 같은 목적을 달성하기 위해서 자연어 처리나 이미지 캡셔닝 분야에서 사용되는 정량적 평가 방법들의 특징에 대해서 분석한 후에 자동 작곡 모델 평가에 실제로 적용해 보았으며, 이를 통해 딥러닝의 다른 분야에서 사용되는 정량적인 평가 방법이 자동 작곡 분야에도 적용 가능한지에 대해 가능성과 한계점을 살펴보았다. 딥러닝을 이용한 자동 작곡에서는 순환 신경망(Recurrent Neural Network, RNN)[Hochreiter. S. 등, 1997 : Mikolov. T. 등, 2010 : Sak. H. 등, 2014 : Sutskever. I. 등, 2014 : Jozefowicz. R. 등, 2015]을 주로 사용한다. 순환 신경망을 이용하기 위해서는 먼저 음악을 구성하고 있는 음표를 순환 신경망이 인식할 수 있는 수치 데이터의 형태로 바꿔야 한다. 기존의 자동 작곡 연구에서는 주로 데이터를 0과 1로 이루어진 이진 벡터(vector)로 변환시켜주는 원-핫 인코딩 방식을 사용하거나[Chu. H. 등, 2016], 자연어 처리 분야에서 단어를 수치 데이터로 변환시켜주는 단어 임베딩 방식을 응용한 음표 임베딩 방식을 사용한다[Bretan. M. 등, 2016 : Wang. Z. 등, 2020 : Wang. Z. 등, 2020]. 원-핫 인코딩 방식과 음표 임베딩 방식 중에서 어떤 방식이 인간이 작곡한 곡과 더 유사한 곡을 생성할 수 있는지를 정량적으로 평가하고자 본 연구에서는 두 가지 방식을 모두 사용하여 음악 데이터를 학습시키고 곡을 생성하였으며, 앞에서 서술한 자연어 처리 등 다른 분야에서 사용되는 정량적 평가 방법을 이용하여 생성된 결과를 비교 평가하였다. 뿐만 아니라 자연어 처리에서 단어 임베딩 외에 문자(character) 임베딩이 가능한 것처럼 자동 작곡에서도 임베딩의 적용 단위(unit)를 세분화시켜서 기존의 음표 임베딩 외에 새로이 마디(bar) 임베딩을 구현하였으며, 자연어 처리 등 다른 분야에서 사용되는 정량적 평가 방법을 이용해서 음표 임베딩의 결과와 비교 평가하였다. 이를 통해서 원-핫 인코딩, 음표 임베딩, 마디 임베딩 중에서 어느 방식이 인간이 작곡한 곡과 더 유사한 음악을 생성하는지를 정량적 평가에 근거해서 확인하였다.

Ⅱ. 관련 연구

본 연구에서는 인간이 작곡한 곡과 생성된 곡사이의 유사성을 비교하되 서로 간에 일치하는 음표의 빈도수뿐만이 아니라 새로 생성된 음표의 배열이 인간이 작곡한 곡과 얼마나 유사한지도 측정한다. 본 연구에서와 같이 인간이 작곡한 곡과 생성된 곡사이의 연관성을 서로 간에 일치하는 음표의 빈도수뿐만이 아니라 음표의 배열 정도까지 비교하는 연구는 없다. 다만, 음의 높이나 음표의 수와 같은 특성(feature)들을 설정하여 학습에 사용된 곡과 생성된 곡사이의 특성의 분포 상태를 비교하는 연구는 있다. 본 장에서는 특성의 분포 상태를 비교하는 기존의 평가 방법 및 딥러닝의 다른 분야에서 사용되는 정량적 평가 방법과 자연어 처리 분야에서 사용되는 단어 임베딩에 대해서 소개한다.

2.1 기존의 정량적 평가 방법

음의 높이나 음표의 수와 같은 특성(feature)들을 설정하고 학습에 사용된 곡과 생성된 곡으로부터 각 특성들을 추출한 후에 특성 값의 분포로부터 각각 확률 밀도 함수를 구한다. 확률 밀도 함수 그래프 간에 서로 겹치는 정도를 의미하는 중복 영역(Overlapped area, OA)과 확률분포의 차이를 계산하는 쿨백-라이블러 발산(Kullback-Leibler Divergence, KLD)를 측정하여 학습에 사용된 곡과 생성된 곡이 얼마나 유사한 지를 평가한다[Yang. L. C 등, 2018].

평가에 사용된 특성들을 살펴보면 다음과 같다.

1) 피치(Pitch) 기반 특성들

① 피치 카운트(PC): 샘플 내의 서로 다른 피치 수이다.

② 피치 클래스 히스토그램(PCH): 피치 클래스 히스토그램은 반음계의 12음계로 피치를 옥타브 독립적으로 표현한 것이다.

③ 피치 클래스 전환 매트릭스(PCTM): 피치 클래스의 전환은 키 감지, 코드 인식 또는 장르 패턴 인식과 같은 작업에 유용한 정보를 포함한다.

④ 피치 범위(PR): 피치 범위는 반음들에서 가장 높은 피치와 가장 낮은 피치를 뺄셈으로 계산한다.
⑤ 평균 피치 간격(PI): 연속되는 두 피치 간의 음정의 평균 값(반음 기준으로)이다.
2) 리듬(Rhythm) 기반 특성들
① 노트 개수(NC): 사용한 음표의 수다. 피치 카운트와 반대로 노트 카운트는 피치 정보를 포함하지 않고 리듬과 관련된 기능이다.
② Average inter-onset-interval(IOI): 연속된 두 음 사이의 시간이다.
③ 노트 길이 히스토그램(NLH): 각 사건의 분류는 기본 단위를 (마디 길이)/96의 길이로 나누어 수행하며, 각 음의 길이는 가장 가까운 길이 범주로 정량화한다.
④ 노트 길이 전환 매트릭스(NLTM): 피치 클래스 전환 매트릭스와 유사하게 노트 길이 전환 매트릭스는 리듬 묘사에 유용한 정보를 제공한다.

전체적인 평가 방법은 두 단계로 구성되어 있다. 첫 번째 단계에서는 학습에 사용된 곡으로 구성된 데이터 세트와 자동 작곡 모델에 의해 생성된 곡으로 구성된 데이터 세트를 준비하여 두 데이터 세트로부터 앞에서 설명한 특성들을 추출한 후, 커널 밀도 추정을 이용하여 각 특성의 확률 밀도 함수를 추정한다. 두 번째 단계에서는 확률 밀도 함수 그래프 간에 서로 겹치는 정도를 의미하는 중복 영역과 확률분포의 차이를 의미하는 쿨백-라이블러 발산을 구하여, 그래프 간에 중복 영역이 크고 쿨백-라이블러 발산이 작을수록 모델에 의해 생성된 곡과 학습에 사용된 곡은 유사하다고 평가한다.

2.2 딥러닝의 다른 분야에서 사용되는 정량적 평가 방법

2.2.1 BLEU(bilingual evaluation understudy)

기계 번역 모델이 생성한 텍스트의 품질을 평가하기 위해서 제시된 정량적인 평가 방법[Papineni. K. 등, 2002]으로, BLEU의 특성을 분석하면 다음과 같다.

1) 기계 번역 모델이 생성한 문장은 후보 문장, 인간이 번역한 문장은 참조 문장이라고 한다.
2) 기계 번역 모델이 생성한 문장의 정확도를 측정하여, 정확도가 클수록 번역 모델의 성능이 우수하다고 평가한다.
3) 중복이 심한 문장이 생성되는 경우

[표 2-1] BLEU 점수 예시 1

후보 문장	the the the the the the the (단어의 종류=1, 단어의 수=7)
참조 문장1	the car is on the road (단어의 종류=5, 단어의 수=6)
참조 문장2	there is a car on the road (단어의 종류=7, 단어의 수=7)

정확도의 기본 개념은 후보 문장을 구성하고 있는 단어가 참조 문장에 자주 등장할수록 정확도가 높다는 것이다. 이러한 측정 방법을 단어 1개당으로 계산했다는 의미로 접두사 유니(uni)를 사용하여 유니그램 정확도(unigram precision)라고 하며, 이를 식으로 표현하면 다음과 같다.

$$\text{유니그램 정확도 } P = \frac{N}{w_t}$$

- N : 참조 문장에 존재하는 후보 문장의 단어 개수
- w_t : 후보 문장을 구성하고 있는 단어의 총 수

이를 예시 1에 적용해보면, 유니그램 정확도 $P = \frac{7}{7} = 1$이 된다. 예시 1처럼 후보 문장을 구성하고 있는 단어가 모두 참조 문장에 등장한다면 정확도는 최댓값인 1이 된다. 하지만, 인공지능 번역 모델이 생성한 문장이 the the the the the the the라고 한다면, 그 누구도 해당 문장이 올바르게 번역된 문

장이라고 동의하지 않을 것이다. 따라서 같은 단어가 연속적으로 나올 때 이를 개선해주어야 하는데, 이를 클리핑(clipping)이라고 하며 개선된 유니그램 정확도는 다음과 같다.

$$\text{개선된 유니그램 정확도 } P = \frac{Count_{clip}}{w_t}$$

- $Count_{clip} = \min(Candidate_{count}, m_{max})$, $Count_{clip}$ = $Candidate_{count}$와 m_{max}의 최솟값
- $Candidate_{count}$: 후보 문장에서 해당 단어의 수
- m_{max} : 후보 문장을 구성하고 있는 단어들과 각각의 참조 문장을 구성하고 있는 단어들이 겹치는 수의 최댓값
- w_t : 후보 문장을 구성하고 있는 단어의 총 수

예시 1의 경우에는 'the'가 7개이므로 $Candidate_{count}$ = 7, m_{max} = 후보 문장을 구성하고 있는 단어는 'the' 1종류 밖에 없으며, 참조 문장1에 두 번, 참조 문장2에 한번 나와서 m_{max}=2이므로, $Count_{clip} = \min(Candidate_{count}, m_{max}) = \min(7, 2) = 2$가 된다. w_t는 7이므로. 이를 개선된 유니그램 정확도에 적용하면 다음과 같다.

$$\text{개선된 유니그램 정확도 } P = \frac{2}{7}$$

4) 지나치게 짧은 문장이 생성되는 경우

[표 2-2] BLEU 점수 예시 2

후보 문장	the car (단어의 종류=2, 단어의 수=2)
참조 문장1	the car is on the road
참조 문장2	there is a car on the road

예시 2에 대해서 유니그램 정확도를 계산하면 다음과 같다.

$$\text{유니그램 정확도} = \frac{\text{Count}_{clip}}{w_t} = \frac{1}{2} + \frac{1}{2} = 1$$

- Candidate_{count} : the = 1, car = 1
- m_{max} : the = 2 (참조 문장1 = 2, 참조 문장2 = 1)
  car = 1 (참조 문장1 = 1, 참조 문장2 = 1)
- Count_{clip} : the = min(1, 2) = 1, car = min(1, 1) = 1
- w_t : 후보 문장을 구성하고 있는 단어의 총 수 = 2

이 경우에도 유니그램 정확도가 1이 나왔지만, 참조 문장과 비교해보면 정확하게 생성된 문장이라고 말하기는 어렵다. 이처럼 지나치게 짧은 문장이 생성될 때, 이를 개선하기 위하여 Brevity Penalty(BP)라는 것을 적용한다.

$$BP = \begin{cases} 1 & \text{if } c > r \\ \exp^{(1-\frac{c}{r})} & \text{if } c \le r \end{cases}$$

[그림 2-1] Brevity Penalty

- c : 후보 문장 sentence의 길이,
- r : 후보 문장과 가장 길이 차이가 작은 참조 문장의 길이

예시 2에 Brevity Penalty(BP)를 적용해보면, 다음과 같다.
c : 후보 문장 sentence의 길이 = 2, r : 참조 문장1의 길이 = 6, 참조 문장2의 길이 = 7이므로, 여기서 r = 6이 된다. BP = $e^{1-\frac{6}{2}} = e^{-2} = \frac{1}{e^2}$이 되고, 1보다 작은 값이 된다.
이제 예시 2에 대해서 길이가 개선된 유니그램 정확도를 계산하면 다음과 같다.

$$\text{개선된 유니그램 정확도 } P = \frac{\text{Count}_{clip}}{w_t} \times BP = 1 \times \frac{1}{e^2}$$

$\frac{\text{Count}_{clip}}{w_t}$만 계산을 했을 때보다 작은 값이 되고 짧은 길이에 대한 벌칙(penalty)을 받고 개선이 되었다고 한다. 인공지능 번역 모델이 생성한 문장이 참조 문장보다 짧을 경우 BP페널티를 적용하여 개선을 하게 되며, 이렇게 개선된 값을 BLEU 점수라고 한다.

$$BLEU = \frac{\text{Count}_{clip}}{w_t} \times BP$$

생성된 문장의 길이가 참조 문장보다 길거나 같다면, BP = 1이 되어서 어떤 벌칙도 받지 않고 BLEU = $\frac{\text{Count}_{clip}}{w_t}$이 된다.

5) n-gram 정확도 / n-gram BLEU 점수

앞서 살펴본 유니그램 정확도의 경우, 생성된 문장(후보 문장)을 구성하고 있는 개별 단어의 빈도수로 접근하는 방법으로서, 단어의 순서를 고려하지 않는 성질을 가지고 있다. 기계 번역 모델이 생성한 문장이 인간이 번역한 문장과 유사하기 위해서는 일치하는 단어의 빈도수도 중요하지만, 단어의 순서도 중요하다.

[표 2-3] BLEU 점수 예시 3

후보 문장	road on is the car the
참조 문장	the car is on the road

위의 예시 3에 대해서 BLEU 점수를 계산하면 다음과 같다.

$$\text{BLEU 점수} = \frac{\text{Count}_{clip}}{w_t} \times BP = \frac{6}{6} \times 1 = 1$$

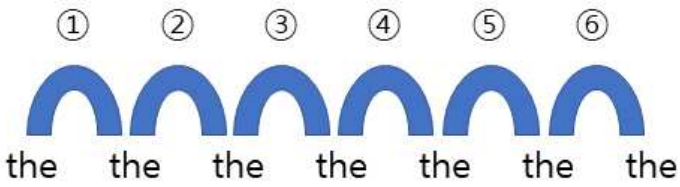
후보 문장과 참조 문장을 비교해보면, 단어의 순서만 다를 뿐 동일한 단어로 구성이 되어 있으므로 BLEU 점수 = 1이 된다. 그러나 이 경우도 기계 번역과 사람의 번역이 완벽하게 일치한다고 생각하기는 어렵다. 그 이유는 바로 단어의 순서가 다르기 때문이다. BLEU 점수 = $\frac{\text{Count}_{clip}}{w_t}$ × BP에서, BP를 제외한 $\frac{\text{Count}_{clip}}{w_t}$ 부분에 이전까지는 유니그램 정확도라고 했으며, 단어의 순서를 무시한 개념이었다.

이제 이를 수정한 n-gram 정확도에 대하여 살펴보고자 한다. n-gram 정확도이란, $\frac{\text{Count}_{clip}}{w_t}$의 값을 계산할 때 개별 단어가 아니라 연속된 단어 단위로 계산을 하는 것으로서, 연속된 2개의 단어 단위로 계산을 하는 경우 2-gram 정확도, 연속된 3개의 단어 단위는 3-gram 정확도, 연속된 4개의 단어 단위로 계산을 하는 경우는 4-gram 정확도라 하며, 이렇게 계산한 BLEU 점수를 n-gram BLEU 점수라고 한다. 2-gram 정확도와 2-gram BLEU 점수(BLEU-2)를 설명할 예시를 들어 보면 다음과 같다.

[표 2-4] BLEU-2 점수 예시 1

후보 문장	the the the the the the the (단어의 종류=1, 단어의 수=7)
참조 문장1	the car is on the road
참조 문장2	there is a car on the road

2-gram 정확도를 계산하기 위해서는 위의 후보 문장과 참조 문장을 다음과 같이 2개의 연속된 단어의 집합으로 변형을 시켜야 한다.



[그림 2-2] 2개의 연속된 단어의 집합으로의 변형

[표 2-5] BLEU-2 점수 예시 2

후보 문장	the the, the the, the the, the the, the the, the the (2개의 연속된 단어 집합의 종류=1, 연속된 단어 집합의 수=6)
참조 문장1	the car, car is, is on, on the, the road
참조 문장2	there is, is a, a car, car on, on the, the road

2-gram 정확도를 구하면 다음과 같다.

$$\text{2-gram 정확도} = \frac{\text{Count}_{clip}}{w_t}$$

- Candidate_{count} : the the = 6
- m_{max} : the the= 0 (참조 문장1 = 0, 참조 문장2 = 0)
- Count_{clip} : the the = min(6, 0) = 0
- w_t : 후보 문장을 구성하고 있는 단어의 총 수 = 6

따라서 2-gram 정확도 = $\frac{\text{Count}_{clip}}{w_t} = \frac{0}{6}$ = 0이 되며, 2-gram BLEU 점수를 구하면,

$$\text{BLEU-2 점수} = \frac{\text{Count}_{clip}}{w_t} \times \text{BP} = 0 \times 1 = 0$$

이 된다. 이처럼 연속된 2개의 단어 단위로 BLEU 점수를 구하면 2-gram BLEU 점수(BLEU-2), 연속된 3개의 단어 단위로 구하면 3-gram BLEU 점수(BLEU-3), 연속된 4개의 단어 단위로 구하면 4-gram BLEU 점수(BLEU-4)라고 한다. 일반적으로 BLEU 점수는 다음과 같이 n-gram BLEU의 평균을 구하여 해당 후보 문장의 BLEU 점수라고 한다.

$$\text{BLEU 점수} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

- N : 일반적으로 1-gram부터 4-gram 까지 사용하며, 따라서 N=4
- P_n : n-gram 정확도
- w_n : n-gram간의 가중치로서 일반적으로 동일하게 설정한다.
- BP : 참조 문장의 길이보다 짧으면 penalty를 준다.

2.2.2 ROUGE(recall-oriented understudy for gisting evaluation)

텍스트 자동 요약, 기계 번역 등 자연어 생성 모델의 성능을 평가하기 위한 지표이며, 텍스트 요약 모델이 생성한 요약본 혹은 번역본을 사람이 미리 만들어 놓은 참조본과 대조해 성능 점수를 계산한다[Lin, C. Y. 2004]. ROUGE의 특성을 분석하면 다음과 같다.

1) 텍스트 요약 모델이 생성한 요약본 혹은 번역본은 후보 문장 요약본, 인간이 미리 만들어 놓은 참조 요약본은 참조 문장 요약본라고 한다.

2) 텍스트 요약 모델이 생성한 요약본과 인간이 미리 만들어 놓은 참조 요약본을 비교하여 정확도(Precision)와 재현율(Recall)을 계산하며, 최종적으로 두 값의 F-measure를 구하여 해당 텍스트 요약 모델의 성능을 평가한다.

[표 2-6] ROUGE 점수 예시

후보 문장 요약본	the the the the the the the
참조 문장 요약본1	the car is on the road
참조 문장 요약본2	there is a car on the road

정확도는 다음과 같이 구한다.

$$\text{정확도 } P = \frac{m_{max}}{w_t}$$

• m_{max} : 후보 문장 요약본을 구성하고 있는 단어들과 각각의 참조 문장 요약본을 구성하고 있는 단어들이 겹치는 수의 최댓값

• w_t : 후보 문장 요약본을 구성하고 있는 단어의 총 수

예시의 경우, 후보 문장 요약본을 구성하고 있는 단어는 'the' 1종류이므로 'the'에 대해서만 계산하면 된다. 'the'는 참조 문장 요약본1에서 2번, 참조 문장 요약본2에서 1번 나왔으므로 m_{max} = 2가되고, w_t = 7이므로

$$\text{정확도 } P = \frac{m_{max}}{w_t} = \frac{2}{7}$$

이 된다. 재현율(Recall)은 참조 문장 요약본을 구성하는 단어 중 몇 개의 단어가 후보 문장 요약본의 단어들과 겹치는지를 보는 점수로서 다음과 같이 구한다.

$$\text{재현율 } R = \frac{n}{w'_t}$$

• n : 참조 문장 요약본을 구성하고 있는 단어들과 후보 문장 요약본을 구

성하고 있는 단어들이 겹치는 수

• w'_t : 참조 문장 요약본을 구성하고 있는 단어의 총 수

참조 문장 요약본 1에 대하여 재현율을 구하면 'the'가 2번 겹치므로,

$$\text{재현율 } R = \frac{n}{w'_t} = \frac{2}{6} = \frac{1}{3}$$

이 되고, 참조 문장 요약본 2에 대하여 재현율을 구하면 'the'가 1번 겹치므로,

$$\text{재현율 } R = \frac{n}{w'_t} = \frac{1}{7}$$

이 된다. 이처럼 참조 문장 요약본이 여러 개 존재하는 경우, 재현율은 최댓값을 구한다. 따라서 예시 1에서의 재현율은 $\frac{1}{3}$이 된다. 정확도 P와 재현율 R의 F-measure를 계산하면[Lin. H. Y, 2004],

$$\text{F-measure} = \frac{(1+\beta^2)(R \times P)}{(R+\beta^2 P)}$$

이고, 여기서 β는 재현율에 얼마나 큰 비중을 주는가를 결정해주는 계수(coefficient)이며 논문에서 제시한 값은 1이다. 그러므로 본 연구에서도 β=1을 적용한다.

정리하면, $\text{F-measure} = \frac{(1+\beta^2)(R \times P)}{(R+\beta^2 P)} = \frac{2RP}{R+P}$가 된다. 이를 예시 1에 적용해서 ROUGE 점수를 구하면,

$$\text{ROUGE 점수} = \text{F-measure} = \frac{2RP}{R+P} = \frac{2\times\frac{1}{3}\times\frac{2}{7}}{\frac{1}{3}+\frac{2}{7}} = 0.3077$$

이 된다. 또한 ROUGE는 참조 문장 요약본이 여러 개 존재하는 경우 정확도와 재현율 모두 최댓값을 구하여 반환한다.

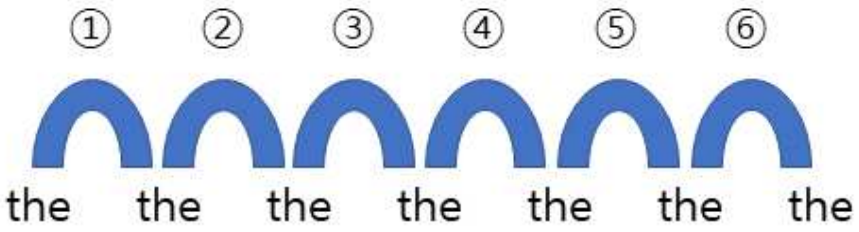
3) n-gram ROUGE 점수

위에서 살펴본 ROUGE 점수는 단어 1개를 기준으로 측정을 한 경우이고, 연속된 2단어 이상에 대해서 정확도와 재현율을 구하여 F-measure를 측정하면 n-gram ROUGE 점수가 된다. 먼저, 2-gram ROUGE 점수(ROUGE-2)를 설명할 예시를 들어 보면 다음과 같다.

[표 2-7] ROUGE-2 점수 예시 1

후보 문장 요약본	the the the the the the the
참조 문장 요약본1	the car is on the road
참조 문장 요약본2	there is a car on the road

2-gram ROUGE 점수를 계산하기 위해서는 앞에서 2-gram BLEU 점수를 계산할 때와 같이 후보 문장 요약본과 참조 문장 요약본을 다음과 같이 2개의 연속된 단어의 집합으로 변형을 시켜야 한다.



[그림 2-3] 2개의 연속된 단어의 집합으로의 변형

[표 2-8] ROUGE-2 점수 예시 2

후보 문장 요약본	the the, the the, the the, the the, the the, the the(2개의 연속된 단어 집합의 종류=1, 연속된 단어 집합의 수=6)
참조 문장 요약본1	the car, car is, is on, on the, the road
참조 문장 요약본2	there is, is a, a car, car on, on the, the road

참조 문장 요약본1과 참조 문장 요약본2의 재현율을 구하면 후보 문장 요약본과 겹치는 단어가 없으므로 각각 0이 되고, 정확도역시 후보 문장 요약본과 참조 문장 요약본 사이에 겹치는 단어가 없으므로 0이 된다. 2-gram ROUGE 점수(ROUGE-2)를 구하면,

$$\text{ROUGE-2} = \text{F-measure} = \frac{2RP}{R+P} = \frac{0}{0}$$

이 되어 계산 불능에 빠지게 된다. 따라서 실제 ROUGE 점수를 구하는 경우, $\frac{2RP}{R+P}$의 분모에 매우 작은 수를 더하여 계신 불능이 되지 않고 0이 되도록 한다. 결론적으로 해당 예시의 ROUGE-2 =0이 된다. 이처럼 연속된 2개의 단어 단위로 ROUGE 점수를 구하면 2-gram ROUGE 점수(ROUGE-2), 연속된 3개의 단어 단위로 ROUGE 점수를 구하면 3-gram ROUGE 점수(ROUGE-3)라 한다.

4) ROUGE-L

앞에서 살펴본 n-gram BLEU 점수나 n-gram ROUGE 점수는 모두 인공지능 모델이 생성한 후보 문장과 인간이 만든 참조 문장에 일치하는 연속된 단어의 수가 많을수록 두 문장 간의 유사도가 더 높다는 것을 의미한다. BLEU 점수는 일반적으로 4-gram까지 구하고, ROUGE 점수에서는 별도의 n 값을 정하지 않는다. ROUGE에서는 n-gram외에 Longest Common Subsequence (LCS)을 이용하여 F-measure와 ROUGE 점수를 구하는데 이것을

ROUGE-L이라 한다. n-gram이 단어 수준에서 연속된 일치를 구하는 개념이라면, LCS는 문장 수준에서 단어의 순서를 반영하는 연속된 일치를 구하는 개념이다. 다음 예시를 통해 n-gram과 LCS의 차이점을 구별해본다.

[표 2-9] ROUGE-L 점수 예시

참조 문장 요약본	police helped the wounded
후보 문장 요약본1	police help the wounded
후보 문장 요약본2	the wounded help police

설명을 위해 n-gram은 ROUGE-2, 즉 n = 2 만 고려한다. 후보 문장 요약본1과 후보 문장 요약본2는 둘 다 the wounded라는 1개의 연속된 두 단어의 일치가 존재한다. 따라서 후보 문장 요약본1과 후보 문장 요약본2의 ROUGE-2는 동일하다. 그러나 문장 수준에서 후보 문장 요약본1과 후보 문장 요약본2의 의미는 다르다. 이것이 n-gram의 한계점이라 할 수 있다. 그렇다면 두 문장 중에서 어느 것이 참조 문장 요약본과 더 유사한가? 두 문장의 주어와 동사, 목적어 등 단어의 순서가 비슷할수록 더 유사하게 나타날 것이다. 후보 문장 요약본1의 경우, 참조 문장 요약본과 문장 수준에서 순서가 일치하는 단어의 수는 police, the, wounded로 3개이다. 후보 문장 요약본2의 경우, 참조 문장 요약본과 문장 수준에서 순서가 일치하는 단어의 수는 the, wounded로 2개이다. 정리하면, 후보 문장 요약본1의 경우 LCS=3이고, 후보 문장 요약본2의 경우 LCS=2가 된다. 이제 각각의 LCS 값을 가지고 정확도와 재현율을 구하면 되는데, 식으로 정리하면 다음과 같다.

$$\text{후보 문장 요약본1의 정확도 } P = \frac{LCS(C,R)}{w_t} = \frac{3}{4},$$

$$\text{후보 문장 요약본2의 정확도 } P = \frac{LCS(C,R)}{w_t} = \frac{2}{4},$$

후보 문장 요약본1에 대한,

$$\text{참조 문장 요약본의 재현율 } R = \frac{LCS(C,R)}{w'_t} = \frac{3}{4},$$

후보 문장 요약본2에 대한,

$$\text{참조 문장 요약본의 재현율 } R = \frac{LCS(C,R)}{w'_t} = \frac{2}{4}$$

가 된다.

- C : 후보 문장 요약본, R : 참조 문장 요약본,
- w_t : 후보 문장 요약본을 구성하고 있는 단어의 총 수,
- w'_t : 참조 문장 요약본을 구성하고 있는 단어의 총 수

각각의 ROUGE-L를 구하면 다음과 같다.

$$\text{후보 문장 요약본1의 ROUGE-L} = \frac{2RP}{R+P} = \frac{3}{4},$$

$$\text{후보 문장 요약본2의 ROUGE-L} = \frac{2RP}{R+P} = \frac{2}{4}$$

후보 문장 요약본1이 후보 문장 요약본2보다 참조 문장 요약본과 더 유사하다고 평가되며, 이는 사람들의 상식에 부합하는 결과이다. 이처럼, ROUGE-L은 n-gram의 한계점을 개선해서 두 문장 간의 유사도를 보다 더 정확하게 평가한다고 할 수 있다.

2.2.3 METEOR

기계 번역(Machine Translation)에 대한 평가를 위해 제시한 정량적인 평가 방법이다[Banerjee, S. 등, 2005]. METEOR의 특성을 분석하면 다음과 같다.

1) 기계 번역 모델이 생성한 문장은 후보 문장, 인간이 번역한 문장은 참조 문장라고 한다.

2) 후보 문장과 참조 문장 사이의 정확도(정확도)와 재현율(Recall)을 계산하며, 최종적으로 두 값의 F-measure를 구하여 해당 기계 번역 모델의 성능을 평가한다.

$$\text{정확도 } P = \frac{m_{max}}{w_t}$$

• m_{max} : 후보 문장을 구성하고 있는 단어들과 각각의 참조 문장을 구성하고 있는 단어들이 겹치는 수의 최댓값
• w_t : 후보 문장을 구성하고 있는 단어의 총 수

$$\text{재현율 } R = \frac{n}{w'_t}$$

• n : 참조 문장을 구성하고 있는 단어들과 후보 문장을 구성하고 있는 단어들이 겹치는 수
• w'_t : 참조 문장을 구성하고 있는 단어의 총 수

METEOR에서의 F-measure는 다음과 같다.

$$\text{F-measure} = \frac{(1+\beta^2)(R \times P)}{(R+\beta^2 P)} = \frac{10RP}{R+9P}$$

(ROUGE에서는 β=1이지만, METEOR에서 β=3이다)

METEOR에서도 BLEU의 Brevity Penalty처럼 페널티가 존재한다.

[표 2-10] METEOR 점수 예시

후보 문장	the man spoke to the audience
참조 문장	the man then spoke to the audience

METEOR에는 chunk라는 개념이 존재하는데, chunk는 후보 문장과 참조 문장에서 인접한 유니그램 세트(일치하는 연속된 단어의 뭉치)로 정의된다. 이

를 예시에 적용하면, 후보 문장에는 참조 문장과 비교하여 서로 일치하는 연속된 단어의 뭉치인 chunk가 the man, spoke to the audience 이렇게 2개가 존재한다. 후보 문장에서 chunk를 구했으면 다음과 같이 페널티를 구한다.

$$\text{Penalty} = 0.5 \times (\frac{c}{v_m})^3$$

- c : chunk의 수, 예시의 경우 c=2가 된다.
- v_m : 두 문장 간에 서로 매치되는 단어의 수, 예시의 경우 v_m=6이다.

$$\text{Penalty} = 0.5 \times (\frac{c}{v_m})^3 = 0.5 \times (\frac{1}{3})^3 = \frac{1}{54}$$

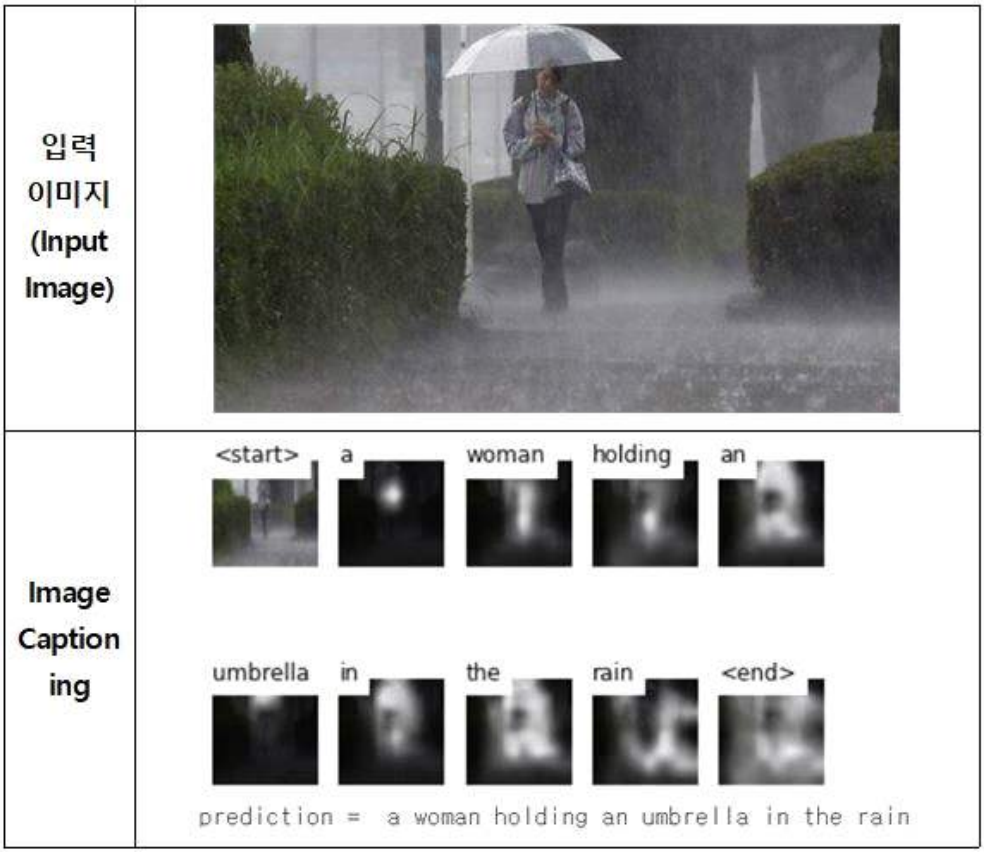
페널티를 구했다면, 다음과 같이 최종적으로 METEOR 점수를 구할 수 있다.

$$\text{METEOR 점수} = \frac{10RP}{R+9P} \times (1 - \text{Penalty}) = \frac{10RP}{R+9P} \times \frac{53}{54}$$

만일 두 문장이 완벽하게 일치한다면 chunk=1이 되고 페널티는 최솟값이 되며, METEOR 점수는 최대가 된다. 반대로 두 문장의 유사도가 낮아질수록 chunk의 값은 증가하며, chunk의 값이 커질수록 페널티는 커지고 METEOR 점수는 작아진다. 이처럼 METEOR는 문장 수준에서의 유사도를 평가할 수 있다.

2.2.4 CIDEr(Consensus-based Image Description Evaluation)

이미지 캡셔닝(Image Captioning)이란 딥러닝 분야의 하나로, 입력으로 어떤 이미지(Image)가 주어졌을 경우, 해당 이미지에 대한 묘사를 자연어의 형태로 생성하는 것이다.



[그림 2-4] 이미지 캡셔닝의 예

CIDEr은 이미지 캡셔닝(Image Captioning) 생성 모델에 대한 정량적인 평가를 위해서 제안된 것이다[Vedantam. R 등, 2015], CIDEr의 특성을 분석하면, 학습 데이터 세트를 구성하고 있는 이미지 주석에서 자주 등장하는 단어는 불용어(Stopword)일 확률이 높기 때문에, 입력 이미지에 대해서 이미지 캡셔닝 모델이 생성한 주석에 학습 데이터 세트에서 자주 등장하는 단어가 많이 포함되어 있다면 해당 모델의 성능이 좋다고는 할 수 없다. 이처럼 많은 이미지에 자주 등장하는 n-gram에 대한 평가의 비중을 낮추기 위해서 다음 식과 같이 TF-IDF(Term Frequency Inverse Document Frequency)를 사용하여 자주 발생하는 n-gram에 대한 가중치를 낮춰준다.

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{w_l \in \Omega} h_l(s_{ij})} log\left(\frac{|I|}{\sum_{I_{p \in I}} min(1, \sum_q h_k(s_{pq}))}\right),$$

n-gram에 대한 CIDEr_n 점수는 이미지 캡셔닝 모델이 생성한 후보 문장 문장과 사람이 주석을 단 참조 문장 사이의 코사인 유사도의 평균을 이용하여 계산한다.

$$\text{CIDEr}_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{\boldsymbol{g^n}(c_i) \cdot \boldsymbol{g^n}(s_{ij})}{\|\boldsymbol{g^n}(c_i)\| \|\boldsymbol{g^n}(s_{ij})\|},$$

CIDEr은 다음과 같이 다양한 길이의 n-gram 점수를 더한 값이다.

$$\text{CIDEr}(c_i, S_i) = \sum_{n=1}^{N} w_n \text{CIDEr}_n(c_i, S_i),$$

CIDEr의 핵심은 이미지 캡셔닝 모델의 성능을 평가할 때, 입력 이미지에 대해서 적절한 주석을 생성하게 한 다음, 생성된 주석에 학습 데이터 세트에서 자주 등장하는 단어가 많이 포함되어 있다면 TF-IDF라는 페널티를 적용해서 모델의 성능을 낮게 평가한다는 것이다.

2.3 단어 임베딩

1.2절에서 살펴본 것처럼 순환 신경망을 이용해서 음악을 학습시키고, 새로운 음악을 생성하기 위해서는 먼저 곡을 구성하는 음표를 순환 신경망이 인식하고 처리할 수 있는 숫자 형태로 바꿔줘야 한다. 수치 데이터로 변환시키는 방법으로는 앞에서 살펴본 것처럼 0과 1로 이루어진 이진 벡터로 변환시켜주는 원-핫 인코딩[Chu. H. 등, 2016] 방식도 있지만, 단어 임베딩이라는 방법도 있다.

원-핫 인코딩 방식에 대해서 예시를 통해서 좀 더 자세하게 살펴보면,

[표 2-11] 과일의 종류

과일	개수
사과	10
오렌지	5
바나나	15

과일의 종류는 범주형 데이터에 해당된다. 첫 번째 과일 10개는 사과라는 과일로 분류할 수 있고, 두 번째 과일 5개는 오렌지라는 과일로 분류할 수 있으며, 마지막 15개의 과일은 바나나라는 과일로 분류할 수 있다는 뜻이다. 원-핫 인코딩은 다음과 같이 표현하고 싶은 단어에 1의 값을 부여하고, 다른 단어에는 0을 부여하는 방식인데, 예시에서는 표현해야 할 과일이 총 3가지이므로 [사과, 오렌지, 바나나]의 순서대로 과일의 종류를 표현하기로 한다. 이제 사과라는 종류를 전체 과일의 범주에서 원-핫 인코딩 방식으로 표현을 해보면, 첫 번째 사과 자리에 1을, 나머지 과일의 자리에는 0을 넣는다면 [1 0 0] 이렇게 표현이 될 것이다. 같은 방식으로 오렌지를 표현하면, 첫 번째 사과 자리에 0, 두 번째 오렌지 자리에 1, 마지막 바나나 자리에 0을 넣으면 [0 1 0]으로 표현이 된다. 이렇게 0과 1로 표현이 된 숫자의 집합을 원-핫 벡터(one-hot vector)라고 한다.

[표 2-12] 원-핫 벡터

원-핫 벡터	과일	개수
[1 0 0]	사과	10
[0 1 0]	오렌지	5
[0 0 1]	바나나	15

범주형 데이터를 원-핫 벡터로 변환시키는 것만으로도 순환신경망을 통해서 처리할 수 있지만, 원-핫 인코딩에는 몇 가지 한계가 존재한다. 첫 번째는 표현해야 할 대상의 수가 늘어날수록 원-핫 벡터의 차원이 커져야하고 이는 컴

퓨터 연산에 있어서 계산 량이 늘어나고, 메모리의 문제를 일으키게 된다. 위에서 살펴본 예시의 경우, 표현해야 할 과일의 종류가 3가지 밖에 없기 때문에, 아래와 같이 3×3 행렬로 전체 원-핫 벡터를 표현할 수 있지만,

$$\text{과일의 종류} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

[그림 2-5] 원-핫 벡터의
표현 1

과일의 종류가 100가지라면, 한 가지 과일을 표현하기 위해서 100차원으로 이루어진 vector를 생성해야 한다. 그리고 전체 원-핫 벡터를 표현하기 위해서는 100×100 행렬이 필요하다. 만일, 범주형 데이터의 범주가 1백만(10^6)가지라면 하나의 범주를 표현하기 위해서는 1백만(10^6)차원으로 이루어진 vector가 필요하며, 전체 원-핫 벡터를 표현하기 위해서는 $10^6 \times 10^6$ 크기의 행렬이 필요한데 이러한 vector를 이용해서 내적(inner product)과 같은 연산을 하려고 하면 계산 량도 매우 크게 늘어나면서, 해당 컴퓨터에서 메모리의 문제를 일으키게 된다. 뿐만 아니라, 원-핫 벡터는 메모리에서 차지하는 용량에 비해서 비효율적이다. 예를 들어 다음과 같이 10 × 10 행렬로 표현이 되는 원-핫 벡터가 있다고 하면,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

[그림 2-6] 원-핫 벡터의 표현 2

의미가 있는 데이터인 1은 전체의 10%이고, 나머지 90%는 아무런 의미도 없는 0으로 채워져 있다. 이처럼 원-핫 벡터는 메모리에서 차지하는 용량에 비해서 쓸모 있는 데이터는 너무나 희박한 형태의 벡터(sparse vector)이고

컴퓨터 메모리상에서 공간적 낭비를 초래한다. 두 번째는 벡터 간에 코사인 유사도(Cosine Similarity)와 같은 연산을 할 수 없는 것이다. 코사인 유사도는 두 벡터 간의 각도의 코사인 값을 이용하여 계산하는 유사도를 의미하는데 두 벡터의 방향이 완전히 동일한 경우에는 1의 값을 가지며, 90° 의 각을 이루면 0, 180° 로 반대의 방향을 가지면 -1의 값을 갖게 된다. 따라서 코사인 유사도의 값이 1에 가까울수록 같은 방향을 향하기 때문에 유사도가 높다고 판단할 수 있다.

[그림 2-7] 코사인 유사도

두 벡터 A, B에 대해서 코사인 유사도는 다음과 같이 구할 수 있다.

$$\text{Similarity} = \cos(\Theta) = \frac{AB}{\|A\|\,\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2}\,\sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

두 벡터 A, B가 0과 1로 이루어진 희소 벡터의 형태라면, 두 벡터간의 내적의 결과는 언제나 0이 되기 때문에 코사인 유사도(Cosine Similarity)와 같은 연산을 할 수 없다. 우리는 음표를 학습시켜서 새로운 곡을 생성하고자 한다. 이를 위해서 음표의 높이와 길이와 같은 데이터를 희소 벡터를 생성하는 원-핫 인코딩 방식으로 숫자화해서 학습을 시킨다면 새로운 곡은 생성이 되지만, 코사인 유사도와 같은 연산이 불가능함으로 인해서 다음과 같은 한계에 부딪히게 된다. 작곡가가 작곡을 할 때는 자신의 생각을 작품에 담는다. 이처럼 작곡가가 의도를 가지고 만든 음악을 딥러닝 모델을 이용하여 학습시킬 때,

연속으로 이어지는 음표에 담겨있는 연관성을 학습할 수 있는 경우와 그렇지 못한 경우가 존재한다면 학습 능력의 차이로 인해서 생성되는 곡의 품질이 다르게 나타날 수밖에 없을 것이다. 그러므로 연속된 음표에 담겨있는 연관성을 파악할 수 있는 형태로 변환시켜 주는 것이 필요하다.

딥러닝을 이용한 자동 자곡에서도 음표간의 관계를 학습할 수 있도록 학습할 수 있도록 원-핫 인코딩 방식 외에 다른 방식으로 음표를 수치 데이터로 변환하기 위해서, 자연어 처리 분야에서 단어를 수치 데이터화는 방식인 단어 임베딩이라는 딥러닝 알고리즘을 응용해서 음표 임베딩(note embedding)[Bretan. M. 등, 2016 : Wang. Z. 등, 2020 : Wang. Z. 등, 2020]을 구현하였다. 원-핫 인코딩 방식이 표현하고자 하는 단어의 수만큼의 차원을 가지는 희소한 벡터를 생성하는 반면에 단어 임베딩 방식은 표현하고자 하는 단어의 수보다 적은 차원이지만 실수(real number)를 사용하여 밀집한 형태의 벡터(dense vector)를 생성한다. 예를 들어 1천 개의 단어가 있을 때, 삽살개라는 단어를 원-핫 벡터로 표현을 하면 다음과 같다.

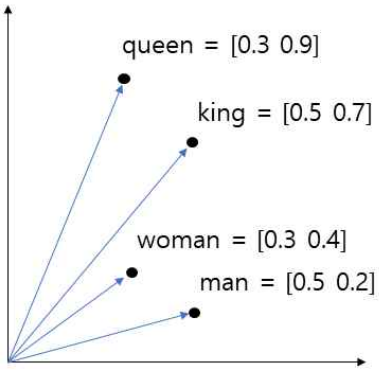
삽살개 = [1 0 0 0 0 0 0 ... 중략 ... 0], 1뒤의 0의 수는 999개

위에서 원-핫 인코딩 방식의 한계점에서 살펴보았듯이 대부분이 0으로 채워져서 비효율적이라 할 수 있다. 만일 1과 0만 사용하는 것이 다음과 같이 실수로 다양하게 표현할 수 있다면 벡터의 크기를 줄이면서도 1천 개의 단어를 전부 표현할 수도 있다.

[표 2-13] 단어 임베딩의 예시

삽살개	[0.11 0.8 0.52 0.76 0.91]
진돗개	[0.33 0.1 0.56 0.79 0.71]
고양이	[0.91 0.35 0.51 0.88 0.1]
돌고래	[0.41 0.27 0.43 0.13 0.9]
원숭이	[0.44 0.23 0.91 0.89 0.2]

원-핫 벡터와는 다르게 모든 차원이 숫자로 채워져 있어서 이러한 형태의 벡터를 밀집 벡터라고 한다. 이처럼 실수로 다양하게 표현할 수 있다면 원-핫 벡터와는 달리 작은 크기의 벡터로도 단어를 효율적으로 표현할 수 있게 되며 이렇게 밀집한 형태의 벡터로 단어를 표현하는 것을 단어 임베딩 혹은 단어의 분산 표현(distributed representation)이라고 한다. 단어 임베딩 방식은 원-핫 인코딩 방식과는 다르게 밀집 벡터를 생성하므로 벡터의 내적을 이용한 코사인 유사도를 계산할 수 있을 뿐만 아니라 두 벡터 간의 거리 측정과 같은 어떤 형태의 연산도 가능하다. 이처럼 벡터간의 연산이 가능하면 유사도뿐만이 아니라, "king - man + woman = queen"과 같은 문제도 그림 8과 같이 벡터 공간에서의 연산을 통해서 풀 수 있으며, 이를 통해 동의어나 반의어를 유추해낼 수 있다.



[그림 2-8] 단어 임베딩

[그림 2-8]에서, 여성을 의미하는 단어들인 queen과 woman의 임베딩 벡터 또는 밀집 벡터 사이의 거리를 구하면,

$$\text{queen} - \text{woman} = \sqrt{(0.3-0.3)^2+(0.9-0.4)^2} = 0.25$$

이번에는 남성을 의미하는 단어들인 king과 man의 임베딩 벡터 사이의 거리를 구하면,

$$\text{king} - \text{man} = \sqrt{(0.5-0.5)^2+(0.7-0.2)^2} = 0.25$$

queen과 woman사이의 거리와 king과 man의 거리가 같으므로,

$$\text{king} - \text{man} = \text{queen} - \text{woman},$$
$$\therefore \text{king} - \text{man} + \text{woman} = \text{queen}$$

이는 남성인 왕에서 남성의 의미를 빼고, 여성의 의미를 더하면 여성인 왕이 된다는 의미이며, “king - man + woman”과 “queen”은 동의어의 관계가 있다는 것을 의미한다. 이처럼 단어 임베딩은 원-핫 인코딩과는 다르게 각 단어에 관계성을 부여할 수 있으며, 이를 통해서 단어 사이의 연관성이나 문맥을 활용할 수 있게 되었고, 그동안 많은 연구자들의 연구에 의해서 다양한 단어 임베딩 방식이 소개가 되었다[Mikolov. T. 등, 2013 : Pennington. J. 등, 2014],

Ⅲ. 모델 설명

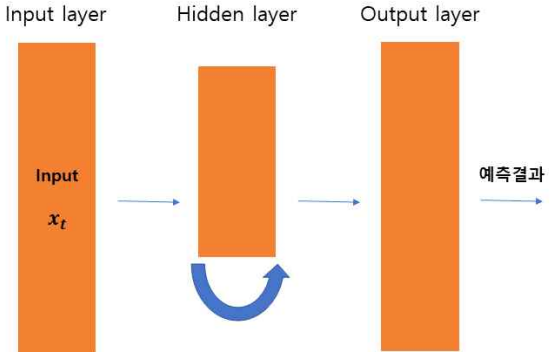
3.1절에서는 곡 평가에 사용할 원-핫 인코딩 방식과 단어 임베딩을 사용하는 자동 작곡 모델에 대하여 소개한다. 3.2절에서는 딥러닝 기반 자동 작곡에서 생성 곡의 정량적 평가에 사용할 평가 방법을 소개한다.

3.1 자동 작곡 모델

본 연구에서는 순환 신경망[Hochreiter. S. 등, 1997 : Mikolov. T. 등, 2010 : Sak. H. 등, 2014 : Sutskever. I. 등, 2014 : Jozefowicz. R. 등, 2015]중에서도 LSTM(Long short Term Memory)을 사용한 언어 모델[Hochreiter. S. 1997 : Alex G. 2008 : Sak. H. 등, 2014 : 김양훈 등, 2016 : 안성만 등, 2017]과 어텐션 mechanism을 적용한 모델을 사용하였다[Bahdanau. D. 등, 2014 : Xu. K. 등, 2015 : Yang. Z. 등, 2016 : Lin. Z. 등, 2017].

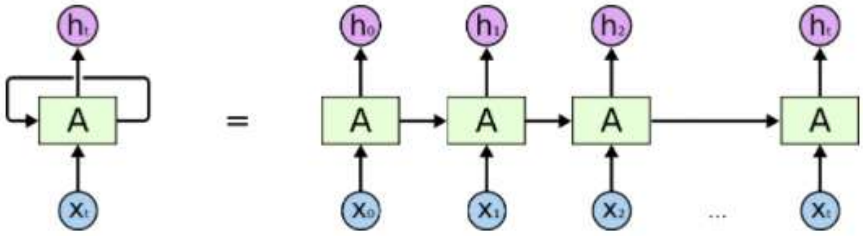
3.1.1 순환신경망 기반 언어 모델(Recurrent Neural Network based Language Model)

순환 신경망의 일반적인 구조는 다음 그림과 같다.

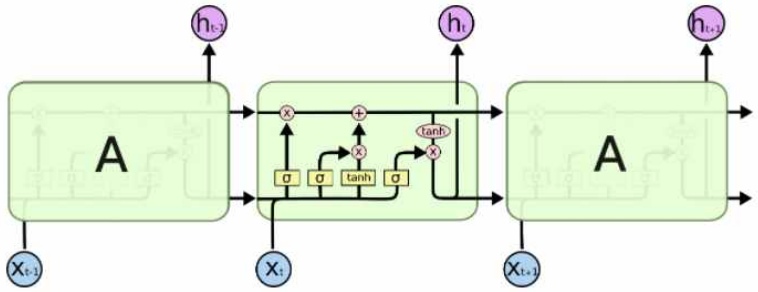


[그림 3-1] 순환 신경망의 일반적인 구조

순환 신경망은 은닉층(Hidden Layer)에서 이전 단계의 정보와 현재의 입력 값을 결합하여 재귀적으로 사용하는 인공 신경망이며, 은닉층으로서 LSTM 층을 사용할 수 있다. LSTM 층은 [그림 3-2]와 같이 재귀적 구조로 되어있으며, 각각의 LSTM 셀은 [그림 3-3]과 같이 Forget Gate, Input Gate, Output Gate로 구성된 3종류의 게이트로 이루어져 있다.



[그림 3-2] 순환 신경망의 재귀적 구조 (Olah, 2015)



[그림 3-3] LSTM 셀의 구조 (Olah, 2015)

각 게이트의 역할은 다음과 같다.

• Forget Gate : 이전 단계(previous state)에서 전달받은 메모리 셀의 값 C_{t-1}을 얼마나 기억할지를 결정한다.

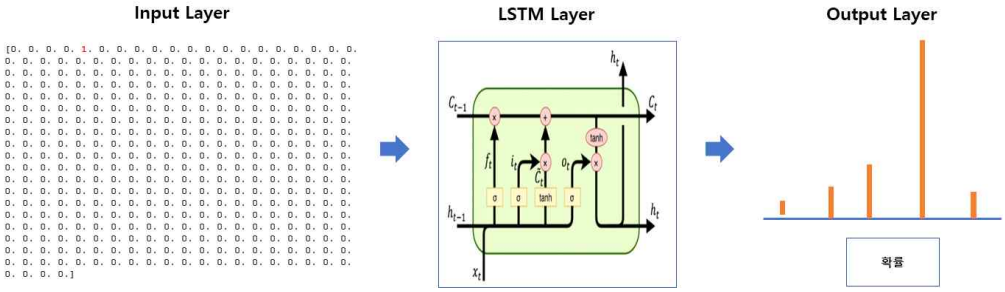
• Input Gate : 현재 단계에서 전달받은 새로운 정보를 얼마나 반영할지를 결정한다.

• Output Gate : 현재 단계에서의 입력과 이전 단계에서의 은닉 상태 h_{t-1}로부터 계산하며, 다음 단계로 얼마나 전달할지를 결정한다. LSTM 셀의 수

식은 아래와 같다[Sak. H. 등, 2014].

$$
\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \\
f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \\
o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \\
m_t &= o_t \odot h(c_t) \\
y_t &= \phi(W_{ym}m_t + b_y)
\end{aligned}
$$

우리 실험에서는 음표 데이터를 입력할 때 원-핫 인코딩을 적용한 순환신경망 언어 모델과 임베딩을 적용한 순환신경망 언어 모델을 사용하였으며, 먼저 원-핫 인코딩을 적용한 모델을 시각화하면 다음과 같다.



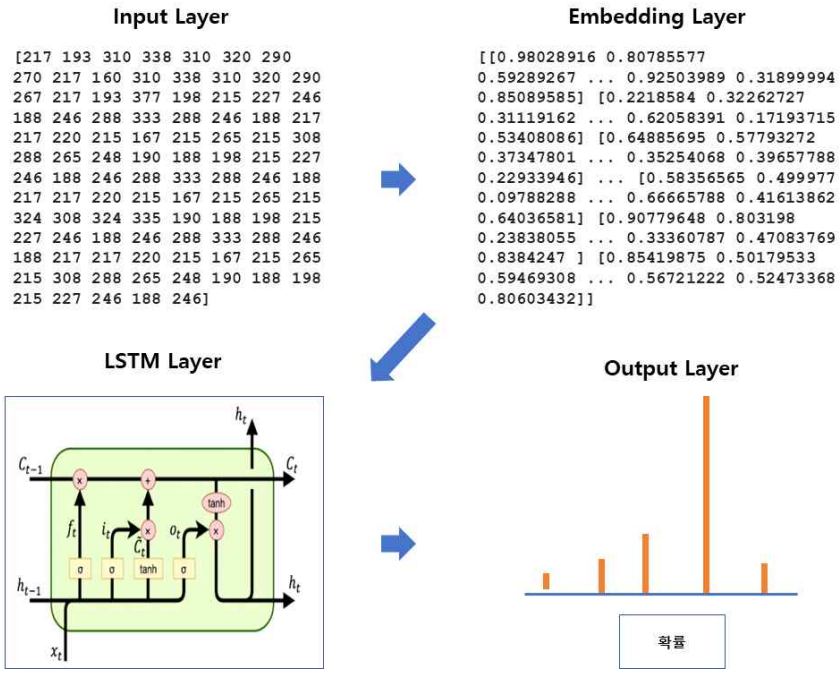
[그림 3-4] 원-핫 인코딩을 적용한 순환신경망 언어 모델

원-핫 인코딩을 적용한 순환신경망 언어 모델을 살펴보면, 총 3개의 층(층)로 이루어진 인공 신경망으로, 먼저 입력층(Input Layer)을 살펴보면 학습 데이터로 사용된 음표를 구성하고 있는 모든 음표와 쉼표에 대해서 종류마다 고유한 인덱스를 부여하고 이를 바탕으로 원-핫 벡터를 생성하여, LSTM 층으로 전달한다. LSTM 층에서는 위에서 살펴본 것처럼 재귀적으로 전달받은 원-핫 벡터를 처리해서 출력층(Output Layer)으로 전달한다. 출력층은 표현하고자 하는 음표의 종류에 해당하는 숫자만큼의 출력 unit으로 구성되는 완전연결 계층(fully connected 층)으로서, LSTM 층의 결과 값과 선형 변환(Linear Transformation)을 통해서 결과 값을 생성한다. 이를 softmax 함수로 처리하면 최종적으로 언어 모델이 예측한 값이 생성되며, 최초에 언어 모델을

학습시킬 때 입력한 정답(target)과 비교하여 손실(loss)을 확정하며, 손실 함수(loss function)로는 언어 모델이 예측한 값과 정답을 비교하는 교차 엔트로피 에러(Cross Entropy Error)를 사용한다.

$$\text{Cross Entropy Error} = -\sum_{i} t_i \log(y_i)$$

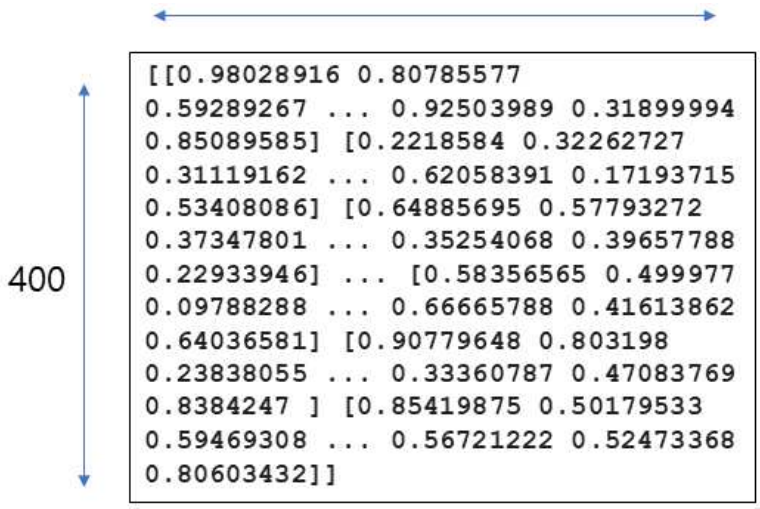
사전에 설정한 손실 함수와 최적화 도구(optimizer)를 이용해서 수렴이 될 때까지 오차 역전파(Error Backpropagation)를 통해서 학습을 진행한다. 이를 통해서 LSTM 셀을 구성하고 있는 각 게이트들의 가중치(weight)를 업데이트(update)하고 언어 모델의 성능을 향상 시킨다. 학습이 종료가 되면, 해당 언어 모델과 학습된 가중치를 이용해서 새로운 곡을 생성하면 된다. 다음으로 단어 임베딩을 적용한 순환신경망 언어 모델을 시각화하면 다음과 같다.



[그림 3-5] 단어 임베딩을 적용한 순환신경망 언어 모델

단어 임베딩을 적용한 순환신경망 언어 모델을 살펴보면, 총 4개의 층(층)

로 구성되어있다. 먼저 입력층을 살펴보면 원-핫 인코딩 방식과는 달리 학습 데이터로 사용된 모든 음의 높이와 길이에 대해서 종류마다 정수(integer)를 부여한 후 이를 그대로 다음 단계인 임베딩 층으로 전달한다. 임베딩 층에서는 전달 받은 정수를 인덱스로 하는 밀집 벡터를 생성한다. 단어 임베딩은 임베딩 층에서 밀집 벡터를 생성하는 알고리즘으로서, 2-3절에서 살펴본 것처럼 단어 임베딩에 의해 생성되는 벡터는 음표와 쉼표의 종류 및 연구자가 설정한 차원을 크기(size)로 하는 행렬의 형태이다. 만일 표현하고자 하는 음표와 쉼표의 종류가 총 400가지이고 연구자가 설정한 벡터의 크기가 32차원이라고 하면 다음과 같은 400 × 32 크기의 행렬로 표현되는 임베딩 벡터가 생성되는 것이다.

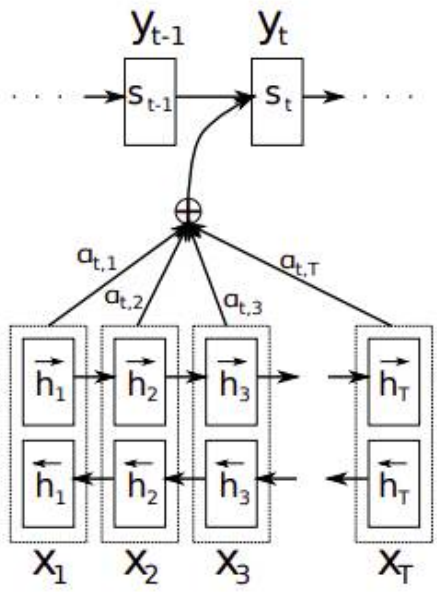


[그림 3-6] 단어 임베딩으로 생성되는 밀집 벡터

임베딩 층에서 생성된 밀집 벡터는 다음 단계인 LSTM 층으로 전달되며 이후의 과정은 원-핫 인코딩을 적용한 순환신경망 언어 모델과 동일하다. 한 가지 다른 점이 있다면 단어 임베딩은 원-핫 인코딩처럼 사람이 수동으로 값을 정하는 것이 아니라, 기계 학습을 통해서 자동으로 수정되고 가장 좋은 결과가 나올 때의 값으로 정해진다.

3.1.2 순환신경망 기반의 어텐션 모델

앞에서 살펴본 임베딩을 적용한 순환신경망 언어 모델을 기본으로 해서 여기에 어텐션 메커니즘(어텐션 mechanism)을 적용해보았다. 어텐션 메커니즘은 기계 번역에서 입력되는 문장이 길어지면 번역의 품질이 떨어지는 문제점을 개선하기 위해 고안된 것으로 다음 그림과 같이 결과를 출력하기 전에 전체 정보 중에서 필요한 부분의 정보를 추가로 반영해서 예측의 정확도를 높이는 알고리즘이다[Bahdanau. D. 등, 2014].



[그림 3-7] 어텐션
메커니즘(Bahdanau. D. 등, 2016)

3.2 평가 방법

우리는 2.2절에서 머신 러닝의 다른 분야에서 사용되는 정량적 평가 방법들의 특성에 대해서 분석해 보았다. 이를 바탕으로 본 연구에 적용이 가능하다고 생각되는 평가 방법도 있지만 적합하지 않은 평가 방법이 있다는 것을 확인할 수 있었다. 적합하지 않은 평가 방법으로는 CIDEr이 있는데, 이미지

캡셔닝 모델의 성능을 평가할 목적으로 개발이 된 CIDEr의 경우 2.2.4절에서 살펴본 것처럼 학습 데이터 세트를 구성하고 있는 이미지 주석에서 자주 등장하는 단어는 불용어일 확률이 높기 때문에, 입력 이미지에 대해서 이미지 캡셔닝 모델이 생성한 주석에 학습 데이터 세트에 서 자주 등장하는 단어가 많이 포함되어 있다면, TF-IDF를 사용하여 해당 단어에 대한 평가의 비중을 낮춰서 해당 모델의 평가 점수가 낮게 나오도록 한다.

평가 모델로서 CIDEr을 곡 생성 평가에 사용하기 어려운 이유는 언어와는 다른 음악의 특성에서 찾아볼 수 있다. 언어에는 불용어가 있지만 음악을 구성하고 있는 음표에는 불용어의 개념이 없이 하나 하나가 다 중요하고 의미를 가지고 있다. 따라서 자동 작곡 모델이 학습 데이터를 구성하고 있는 노래들을 학습해서 새로운 음표들의 배열을 생성한다고 해도 결국 생성된 곡을 구성하는 음표의 종류는 학습 데이터를 구성하고 음표의 일부이며 부분 집합과 전체 집합의 관계이다. 바뀌는 것은 음표의 배열일 뿐 학습데이터에 없는 새로운 음표를 만드는 것은 아니기 때문이다. 그러므로 어떤 자동 작곡 모델로 평가를 하더라도 생성된 곡을 CIDEr로 평가하면 0에 가까운 값이 나올 것이고 이는 곡의 품질을 평가하는 방법으로는 적합하지 않다. 따라서 본 연구에서는 CIDEr을 정량적 평가 방법으로 사용하지 않기로 한다.

그 외에 우리는 2.2절에서 살펴본 나머지 평가 방법들은 곡 생성 평가에 적용해볼 만한 가치가 있기에 본 연구에서 적용해본다. 좀 더 구체적으로 살펴보면, BLEU의 경우 기본 값(default)이 4-gram까지 구해서 기하 평균을 구한 다음 해당 결과를 BLEU 점수로 사용하지만 본 연구에서는 기하 평균을 사용하지 않고, 1-gram부터 4-gram까지 각각의 점수를 구해서 곡 평가에 사용한다. 그 이유는 2-gram이나 3-gram, 4-gram에서 참조 문장과 일치하는 연속된 단어의 배열이 나타나지 않아서 0이 나오는 경우, 0이 존재하는 상황에서 1-gram부터 4-gram까지 전체에 대한 기하 평균을 구하면 역시 0이 나오게 된다. 다음의 예를 살펴보자.

[표 3-1] BLEU 점수 예시

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU 점수
A	0.7	0	0	0	0
B	0.8	0.45	0.2	0	0

생성된 곡을 서로 비교 평가하려고 하는데 위의 예시와 같은 경우라면, A와 B는 분명히 다른 품질이지만 BLEU 점수는 동일하게 0이 나오므로 서로간의 품질을 비교할 수 없다. 따라서 기하 평균 값을 기본 값으로 하는 BLEU 점수로 곡을 평가하기에는 적합하지 않을 수 있다. ROUGE의 경우에는 3-gram까지 단어 수준에서의 겹침을 평가함과 동시에 문장 수준에서의 유사도를 평가하기 위해서 ROUGE-L을 동시에 사용하기로 한다. METEOR의 경우에는 해당 평가 방법의 특성상 단어 수준과 문장 수준에서의 평가를 함께 구현하고 있으므로 그대로 사용하기로 한다. 정리하면 본 연구에서 사용하는 정량적 평가 방법은 3종류 9가지이며, 다음 표와 같다.

[표 3-2] 정량적 평가 방법 정리

BLEU				ROUGE				METEOR
BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	METEOR

IV. 연 구 결 과

4장에서는 원-핫 인코딩 방식과 단어 임베딩 방식의 차이점뿐만이 아니라 단어 임베딩의 경우에도 음표 하나하나를 기본 단위로 해서 자연어의 단어처럼 사용하는 경우와 악보상의 마디(bar)를 기본 단위로 해서 자연어의 단어처럼 사용하는 경우로 나누어서 실험하고 각각의 차이점을 3.2절에서 정한 정량적 평가 방법으로 평가한다.

4.1 실험 데이터

4.1.1 데이터 세트의 구성

본 실험에서는 미디 파일 1,034곡으로 구성된 노팅햄 데이터 세트를 이용하여 다음 표와 같이 학습용과 평가용으로 나누어 사용하였다.

[표 4-1] 데이터 셋의 구성

미디 파일 1,034곡	
학습용 데이터 세트	평가용 데이터 세트
828곡(80%)	206곡(20%)
학습용	평가용

데이터를 학습용과 평가용으로 분리하는 이유는 생성된 곡에 대한 정량적 평가에서 모방(copy)에 의한 고득점을 방지하기 위해서다. 학습용 데이터를 그대로 평가용으로 사용한다면, 자동 작곡 모델이 학습용 데이터를 그대로 모방해서 곡을 생성할 경우 생성된 곡의 정량적 평가 점수는 어떤 평가 방법을 사용해도 1에 가까운 값이 나올 것이고, 해당 모델은 사람이 작곡한 곡과 유사하게 곡을 생성한다고 해석을 하게 될 것이다. 우리의 목표는 인간이 작곡한 곡과 유사한 곡을 생성하는 것이지 모방이 아니다. 따라서 모방에 의한 고득점을 막기 위해서 인간이 작곡하되 학습에 사용되지 않은 데이터와 유사도를 평가하도록 설계하였다.

4.1.2 미디 파일을 텍스트 데이터로 전환

음표를 기계 번역과 같은 자연어 처리에서의 단어처럼 사용하면 단어 임베딩을 구현할 수 있는 장점이 있기 때문에 다음과 같이 텍스트 데이터로 전환하였다.

1) 사용한 라이브러리

음표에 대해서 전 처리 및 시각화를 할 수 있는 파이선(python) 라이브러리인 music21을 이용하였다.

2) 기본 원리 소개

① 음계의 이름과 옥타브를 숫자화 한다. 〈예〉 C4 = 60, C#4 = 61

② 음의 길이를 ①에서 숫자화한 음계의 이름 및 옥타브와 '_' 로 이어서 숫자로 표현한다. 〈예〉 60_0.5 C4 8분 음표, 61_1.0 C#4 4분 음표

③ 쉼표(Rest)에 대해서 독립적인 기호 r를 부여한다.
〈예〉 r_1.0 4분 쉼표

④ 화음의 경우 별표(*)를 이용하여 단음을 연결한다.
〈예〉 60*61_0.5 C4 C#4 16분 음표 2개

⑤ 단음, 쉼표, 화음은 쉼표(,)를 이용하여 구분한다.
〈예〉 r_3.0,60*61_0.5,60_0.5

⑥ Time Signature(박자)는 끝에 'I'를 붙인다. 〈예〉 3/4I

⑦ 각 마디(bar)들은 ' '(white space)로 구분한다.
〈예〉 3/4Ir_2.0,60_1.0 3/4I61_2.0,60_1.0

3) 수치형 데이터로 변환하는 방식에 따른 학습용 데이터 세트의 생성

원-핫 인코딩 방식과 음표 임베딩 방식의 경우 음표를 기본으로 하지만, 마디 임베딩(bar embedding) 방식의 경우 마디가 데이터의 기본 단위이다. 따라서 원-핫 인코딩과 음표 임베딩의 경우에는 학습용 미디 파일 828곡을 텍스트 데이터로 전환시킬 때 각각의 미디 파일에 대해서 마디 구분 없이 음표만으로 구성된 학습용 데이터 세트를 만들고, 마디 임베딩의 경우에는 마디로 구분되는 학습용 데이터 세트를 만든다.

① 음표 단위 학습용 데이터 세트의 예시

```
74_0.5 71_0.5 74_0.5 77_0.5 74_0.5 83_0.5 81_0.5 83_0.5
84_1.0 72_1.0 72_0.5 73_0.5 74_0.5 75_0.5 76_0.5 72_0.5
76_0.5 79_0.5 84_0.5 79_0.5 76_0.5 72_0.5 74_1.0 74_1.0
74_2.0 74_0.5 71_0.5 74_0.5 77_0.5 74_0.5 81_0.5 79_0.5
77_0.5 76_1.0 72_1.0 72_0.5 73_0.5 74_0.5 75_0.5 76_0.5
72_0.5 76_0.5 79_0.5 84_0.5 79_0.5 76_0.5 72_0.5 74_1.0
74_1.0 74_2.0 74_0.5 71_0.5 74_0.5 77_0.5 74_0.5 83_0.5
81_0.5 83_0.5 84_1.0 72_1.0 72_0.5 77_0.5 78_0.5 79_0.5
```

[그림 4-1] 음표 단위 학습용 데이터 세트

② 마디 단위 학습용 데이터 세트

```
4/4|74_0.5,71_0.5,74_0.5,77_0.5,74_0.5,83_0.5,81_0.5,83_0.5
4/4|84_1.0,72_1.0,72_0.5,77_0.5,78_0.5,79_0.5
4/4|72_0.5,69_0.5,72_0.5,76_0.5,81_0.5,76_0.5,72_0.5,69_0.5
4/4|71_0.5,72_0.5,74_0.5,72_0.5,71_0.5,69_0.5,68_0.5,69_0.5
4/4|68_0.5,64_0.5,67_0.5,71_0.5,67_0.5,77_0.5,76_0.5,74_0.5
4/4|72_0.5,71_0.5,69_0.5,68_0.5,69_0.5,64_0.5,69_0.5,71_0.5
4/4|72_0.5,69_0.5,72_0.5,76_0.5,81_0.5,76_0.5,72_0.5,69_0.5
4/4|71_0.5,72_0.5,74_0.5,72_0.5,71_0.5,69_0.5,68_0.5,69_0.5
4/4|68_0.5,64_0.5,67_0.5,71_0.5,67_0.5,77_0.5,76_0.5,74_0.5
```

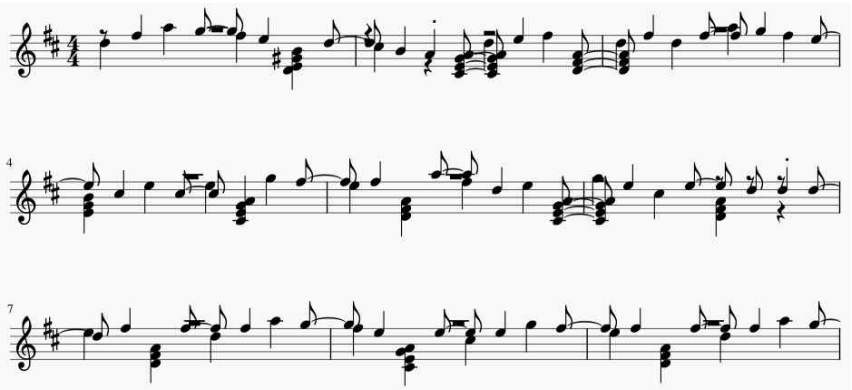
[그림 4-2] 마디 단위 학습용 데이터 세트

4) 평가용 데이터 세트

학습용 데이터는 방식에 따라 적합하도록 음표용과 마디용으로 각각 만들지만, 곡 평가를 할 때는 음표 단위로 학습을 한 경우와 마디 단위로 학습을 한 경우를 비교하여 평가를 해야 하므로 공통된 평가용 데이터가 필요하다. 마디도 음표로 구성이 되어있으므로 마디와 음표의 교집합은 음표이다. 따라서 평가용 데이터 세트는 마디 구분 없이 음표만으로 구성하여 사용한다.

4.1.3 실제 악보에 적용

1) 실제 악보



[그림 4-3] 실제 악보

2) 전환된 텍스트 데이터

```
74_1.0 78_1.0 81_1.0 79_1.0 78_1.0 76_1.0 68*71*62*64_1.0 74_1.0 73_1.0 71_1.0 69_0.5 61*64*67_1.0 74_1.0
76_1.0 78_1.0 62*66*69_1.0 74_1.0 78_1.0 74_1.0 78_1.0 81_1.0 79_1.0 78_1.0 76_1.0 64*67*71_1.0 73_1.0
76_1.0 73_1.0 76_1.0 61*64*67*69_1.0 79_1.0 78_1.0 76_1.0 78_1.0 62*66*69_1.0 81_1.0 78_1.0 74_1.0 76_1.0
61*64*67*69_1.0 79_1.0 76_1.0 73_1.0 76_1.0 62*66*69_1.0 74_0.5 r_0.5 74_0.5 76_1.0 78_1.0 62*66*69_1.0
78_1.0 74_1.0 78_1.0 81_1.0 79_1.0 78_1.0 76_1.0 61*64*67*69_1.0 76_1.0 73_1.0 76_1.0 79_1.0 78_1.0 76_1.0
```

[그림 4-4] 텍스트 데이터

4.2 실험 방법

4.2.1 실험 모델의 종류

모델이 복잡할수록 학습의 대상이 되는 가중치(Trainable Parameter)의 수가 많아지는데, 연산 량이 늘어나는 대신에 데이터의 특성을 보다 자세하게 분석할 수 있으며 학습 역량이 커진다고 할 수 있다, 머신 러닝에서는 모델이 복잡하다고 결과가 항상 좋은 것은 아니다. 모델의 구조가 복잡하더라도 학습 데이터가 단순하다면 오히려 과적합(overfitting)이 발생해서 학습의 결과가 좋지 않을 수 있기 때문에, 데이터의 상태와 모델의 복잡도가 서로 맞아야만

좋은 결과를 기대할 수 있다. 우리가 사용하는 학습 데이터에 적합한 모델을 정확하게 알 수 없기 때문에 다음 표와 같이 총 14가지 모델을 가지고서 곡 생성에 대한 비교 평가를 진행한다.

[표 4-2] 실험 모델의 종류

원-핫 인코딩	LSTM 층 1개	LSTM 층 출력 unit=100
		LSTM 층 출력 unit=512
음표 임베딩	LSTM 층 1개	LSTM 층 출력 unit=100
		LSTM 층 출력 unit=512
	LSTM 층 2개	LSTM 층 출력 unit=100
		LSTM 층 출력 unit=512
	어텐션 모델	LSTM 층 출력 unit=100
		LSTM 층 출력 unit=512
마디 임베딩	LSTM 층 1개	LSTM 층 출력 unit=100
		LSTM 층 출력 unit=512
	LSTM 층 2개	LSTM 층 출력 unit=100
		LSTM 층 출력 unit=512
	어텐션 모델	LSTM 층 출력 unit=100
		LSTM 층 출력 unit=512

원-핫 인코딩 모델의 경우 원-핫 인코딩 방식의 특성상 희소벡터를 생성하기 때문에 계산 량이 단어 임베딩 방식보다 많이 늘어나기 때문에 컴퓨터의 메모리 문제를 발생시킨다. 이로 인해서 좀 더 복잡한 모델에 해당하는 LSTM 층 2개를 사용하거나, 어텐션 메커니즘이 구현된 모델은 메모리 문제로 인해서 실험에 사용하지 못했다. 그러므로 본 실험에서 원-핫 인코딩 방식과 음표 임베딩 방식의 차이를 비교 평가하는 실험은 LSTM 층 1개를 사용한 순환신경망 언어 모델을 이용하여 진행하고, 음표 임베딩 방식과 마디 임베딩 방식의 비교 평가는 LSTM 층 1개, LSTM 층 2개, 어텐션이 구현된 모델을 이용하여 진행한다.

4.2.2 학습 조건

optimizer	learning rate	epoch	early stopping	patience	validation_ split	Embedding layer output unit	batch size
RMSprop	0.002	300	True	10	0.1	100	512

[그림 4-5] 실험 모델의 학습 조건

학습 시 학습 횟수인 에폭(epoch)은 300번으로 설정하되, 과적합을 방지하기 위하여 조기 종료(early stopping) 알고리즘을 사용하였으며, 조기 종료 기능을 구현하기 위해서 학습 데이터 세트의 10%를 검증용 데이터 세트(validation set)로 활용하여 조기 종료의 기준으로 사용하였다. 임베딩 레이어의 출력 크기는 100차원으로 설정하였으며, 미니 배치(mini batch)의 크기는 512로 설정하였다. 학습은 구글에서 제공하는 무료 클라우드 개발 환경인 코랩(Colaboratory)과 코랩에서 제공하는 무료 하드웨어 가속기(GPU)인 T4와 P100을 활용하여 진행하였다.

4.3 평가의 기준

우리의 목적은 생성 곡의 품질이 얼마나 좋은 지를 정량적으로 평가하는 것이며, 정확하게 평가하기 위해서는 평가의 기준이 무엇보다 중요하다. 음악은 인간이 듣기 좋으라고 만든 것이다. 그러므로 평가의 기준은 인간일 수밖에 없다. 다만 인간이 듣기 좋다는 것을 정성적으로 접근하기보다는 정량적으로 접근하고자 한다. 우리는 인간이 작곡한 곡을 평가의 기준으로 설정해서 자동 작곡 모델별로 인간이 작곡한 곡과 얼마나 유사하게 곡을 생성하는지에 대한 비교 평가를 진행한다. 인간이 작곡한 곡을 평가의 기준으로 설정한 이유는 아직까지 가장 듣기 좋은 노래는 인간이 만든 곡이기 때문이다.

그러므로 우리의 목적에 맞는 정량적인 평가 방법이 필요한데 2.2절과 3.2절에서 살펴본 자연어 처리 분야에서 사용되는 정량적인 평가 방법이 대안이 될 수 있다. 그 이유는 처리 분야에서 사용되는 정량적인 평가 방법들이 바로 인간이 만든 문장과 얼마나 유사한가를 평가하기 때문이다.

4.4 실험 결과 및 분석

4.4.1 곡 생성

다음 표와 같이 인코딩 방식 및 모델 별로 각각 50곡씩 생성하였다.

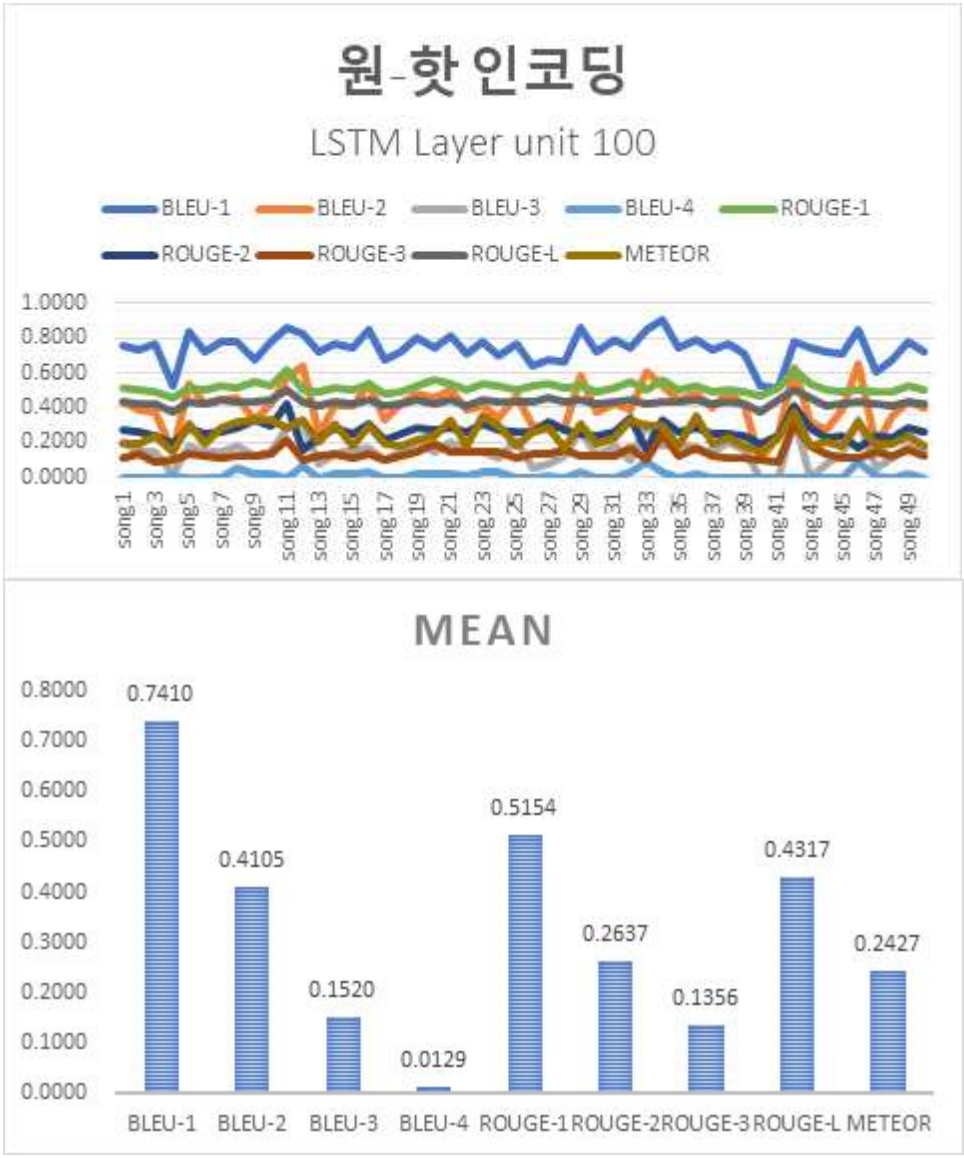
[표 4-3] 실험 모델에 따른 곡 생성 결과

원-핫 인코딩	LSTM 층 1개	LSTM 층 출력 unit = 100	50곡, 음표 160개/1곡
		LSTM 층 출력 unit = 512	50곡, 음표 160개/1곡
음표 임베딩	LSTM 층 1개	LSTM 층 출력 unit = 100	50곡, 음표 160개/1곡
		LSTM 층 출력 unit = 512	50곡, 음표 160개/1곡
	LSTM 층 2개	LSTM 층 출력 unit = 100	50곡, 음표 160개/1곡
		LSTM 층 출력 unit = 512	50곡, 음표 160개/1곡
	어텐션 모델	LSTM 층 출력 unit = 100	50곡, 음표 160개/1곡
		LSTM 층 출력 unit = 512	50곡, 음표 160개/1곡
마디 임베딩	LSTM 층 1개	LSTM 층 출력 unit = 100	50곡, 마디 44개/1곡
		LSTM 층 출력 unit = 512	50곡, 마디 44개/1곡
	LSTM 층 2개	LSTM 층 출력 unit = 100	50곡, 마디 44개/1곡
		LSTM 층 출력 unit = 512	50곡, 마디 44개/1곡
	어텐션 모델	LSTM 층 출력 unit = 100	50곡, 마디 44개/1곡
		LSTM 층 출력 unit = 512	50곡, 마디 44개/1곡

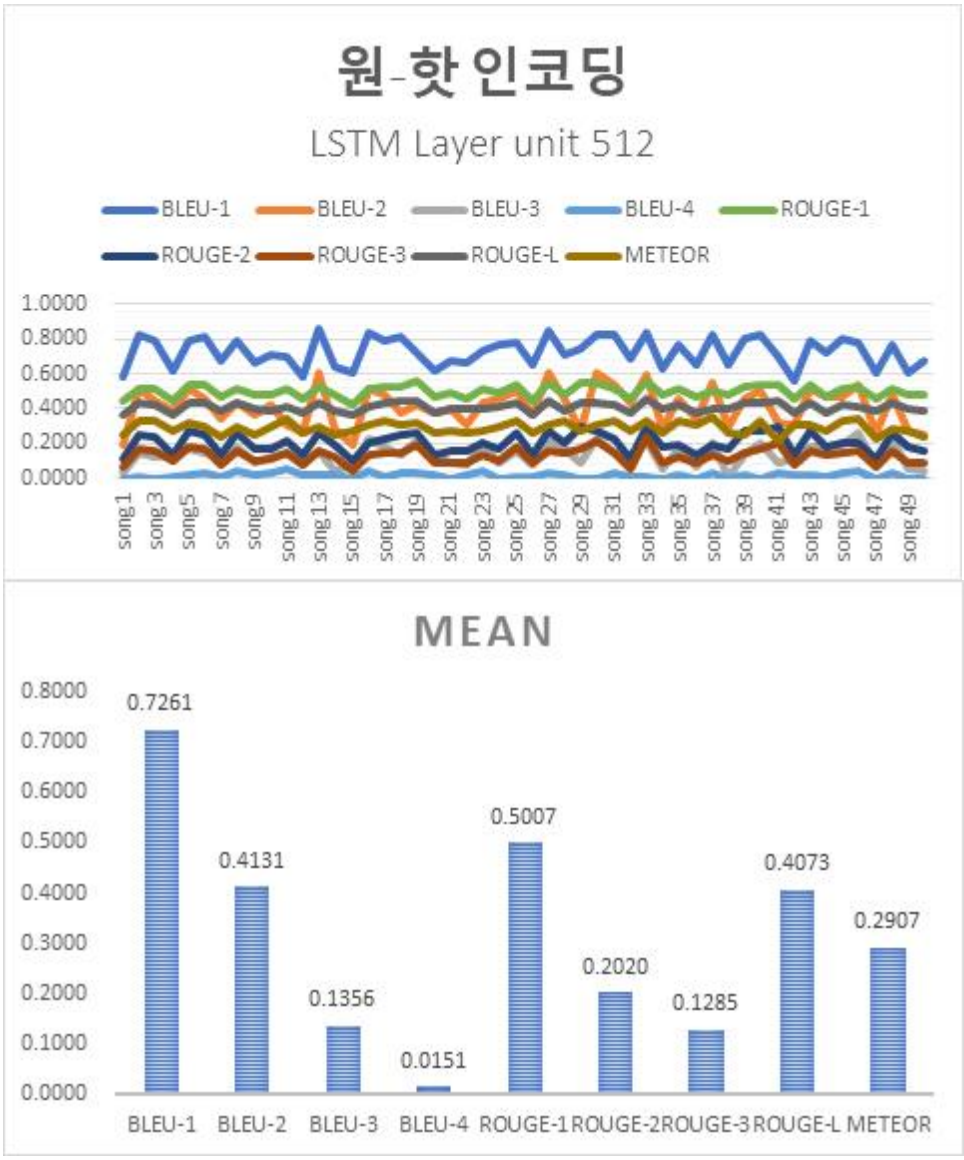
4.4.2 정량적 평가 결과 및 분석

정량적 평가는 14개 실험 모델 별로 각각 생성한 50곡에 대한 평균값으로 진행한다. 각각의 평가 결과를 시각화한 자료는 1)과 같고, 이를 바탕으로 2)와 3)에서 평가 목적 별로 결과를 분석한다.

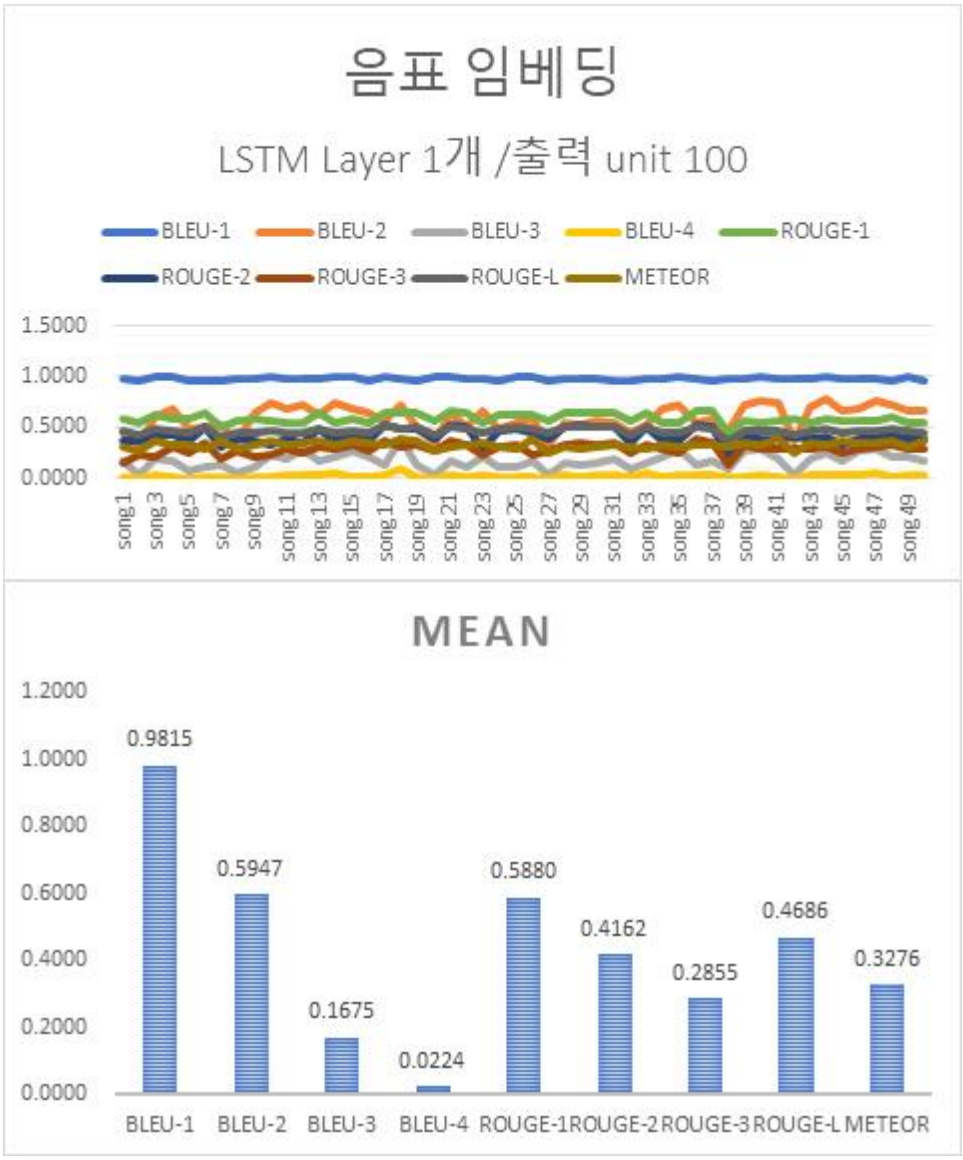
1) 실험 모델 별 평가 결과 자료



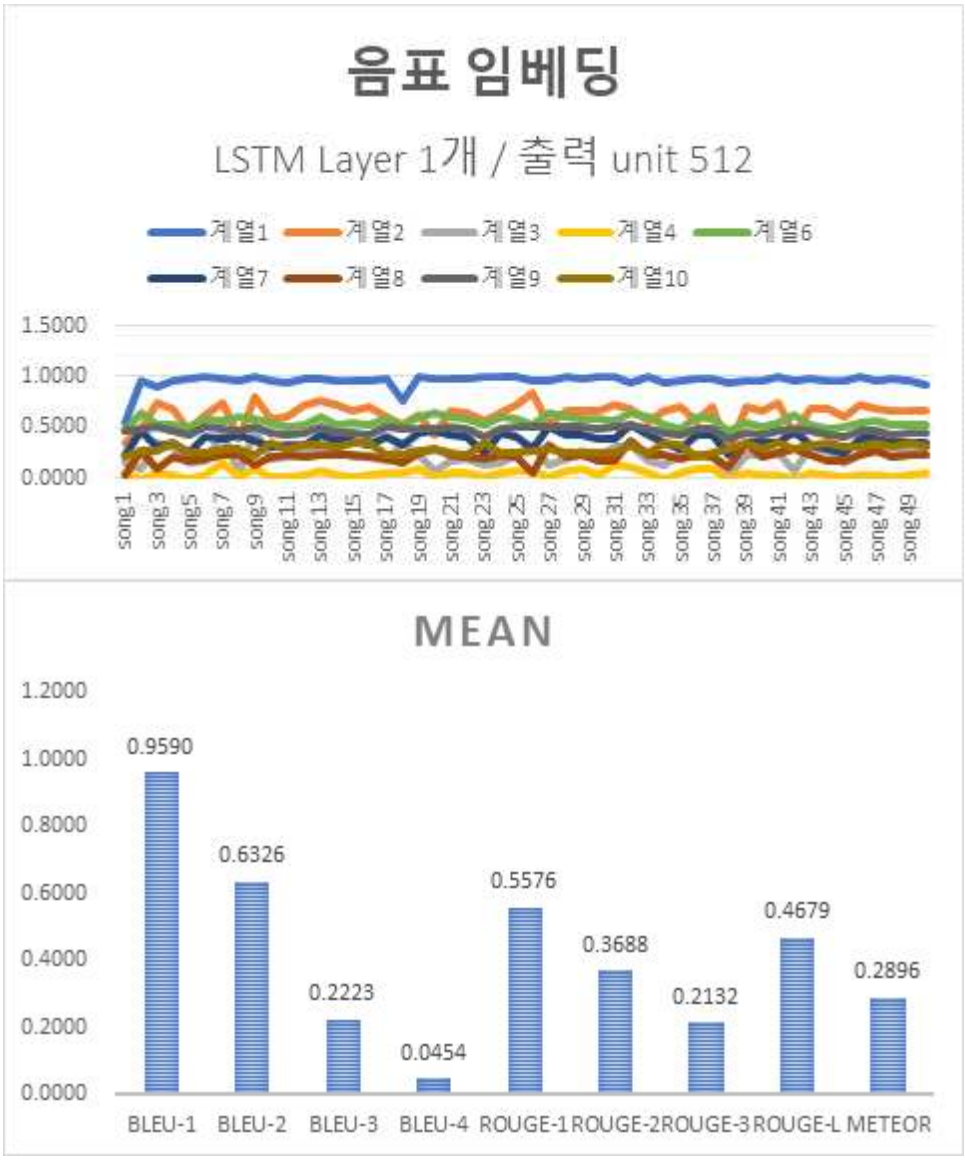
[그림 4-6] 실험 모델 별 평가 결과 자료 1



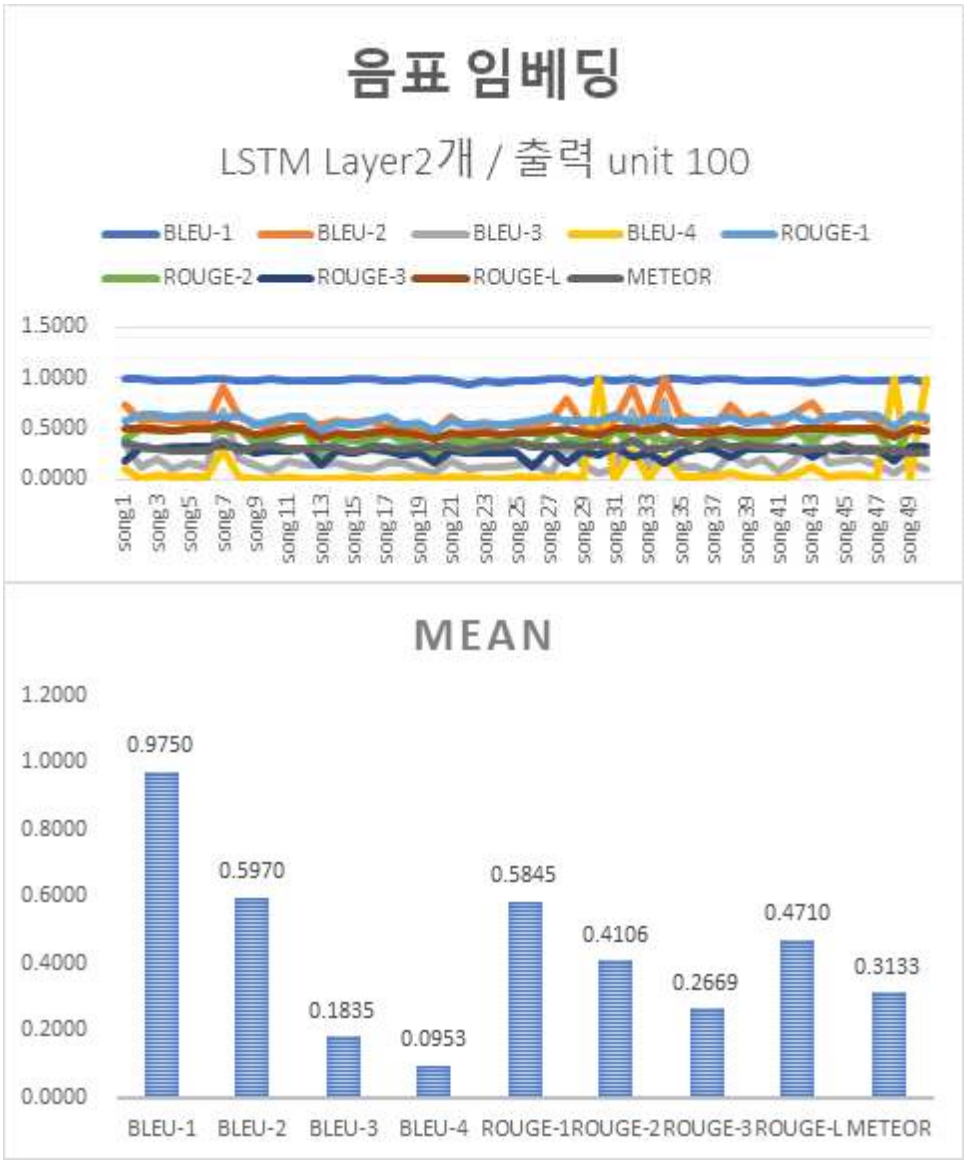
[그림 4-7] 실험 모델 별 평가 결과 자료 2



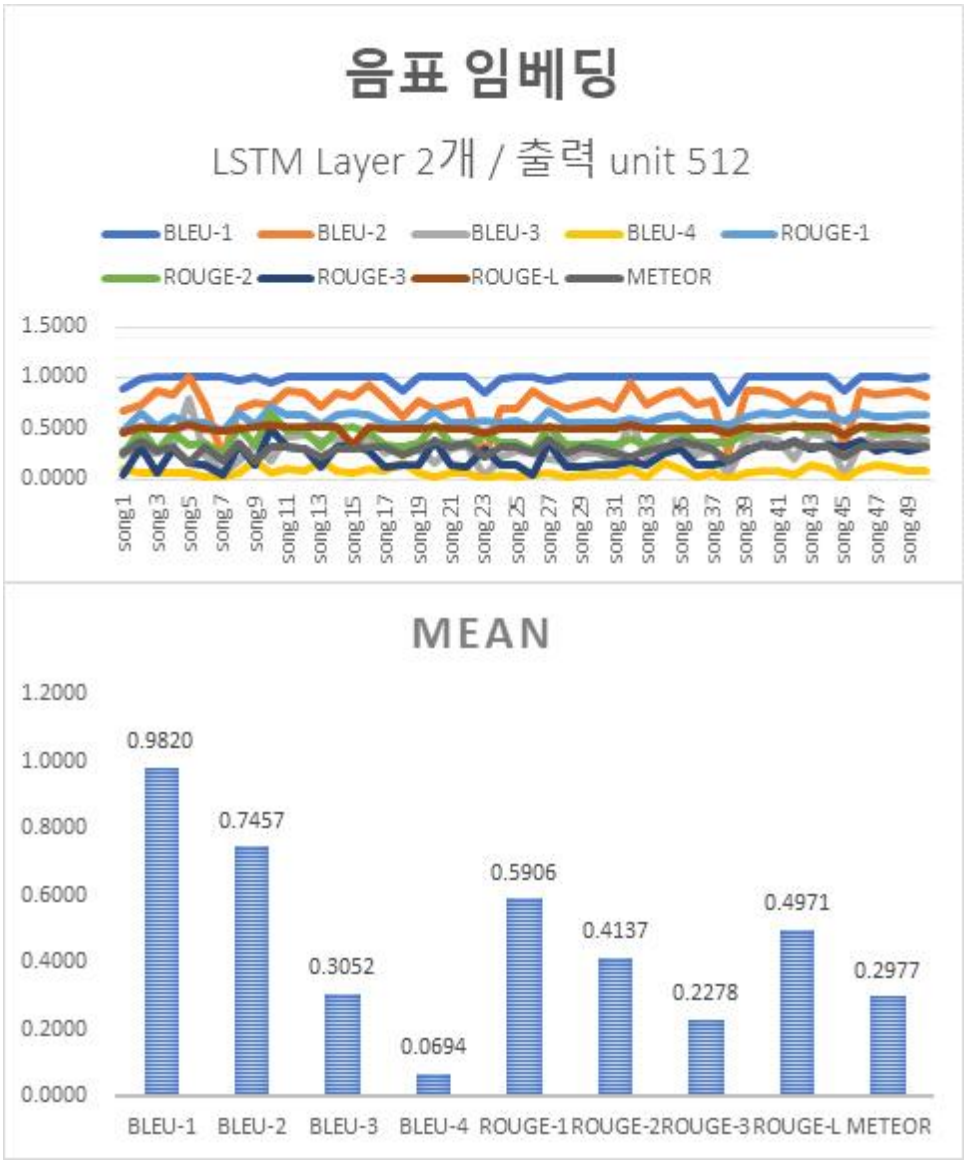
[그림 4-8] 실험 모델 별 평가 결과 자료 3



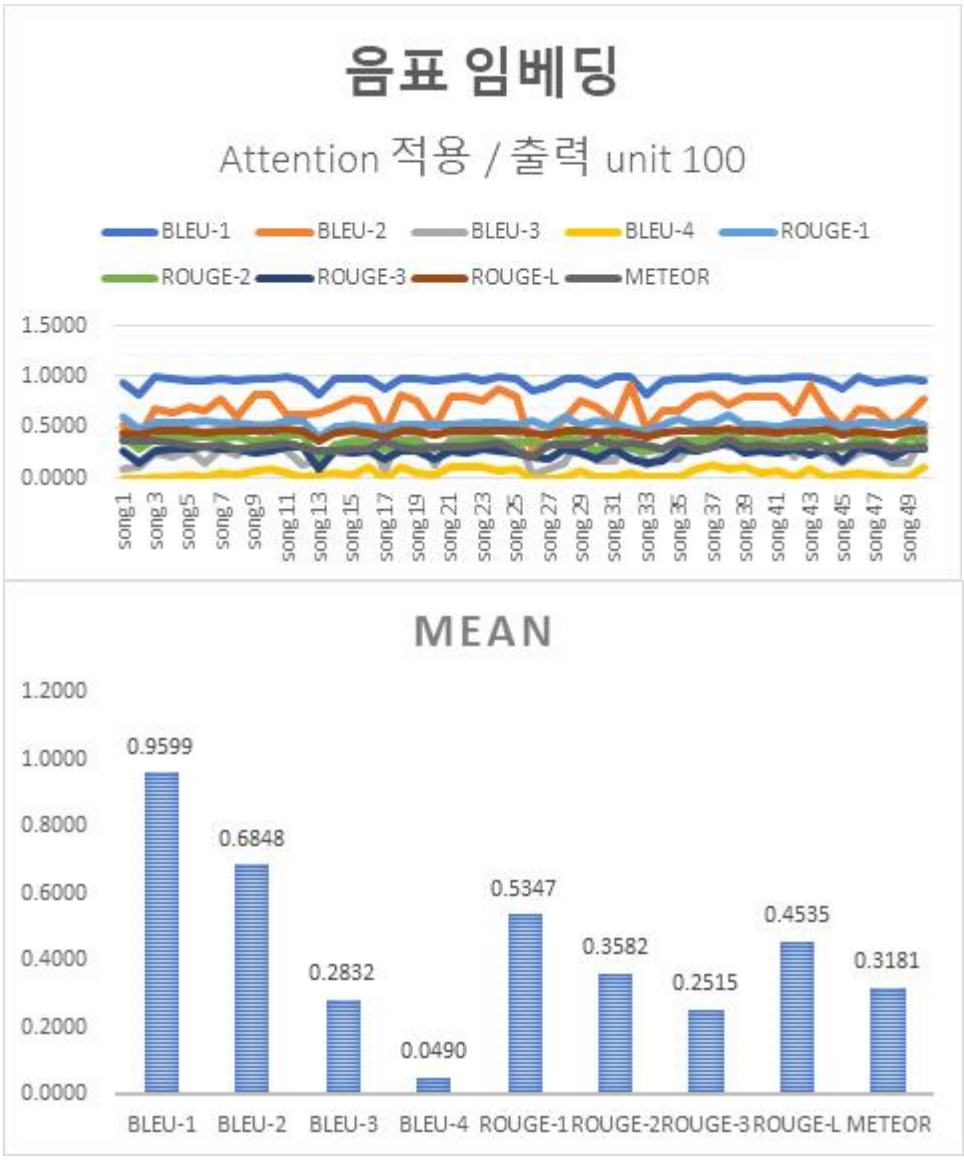
[그림 4-9] 실험 모델 별 평가 결과 자료 4



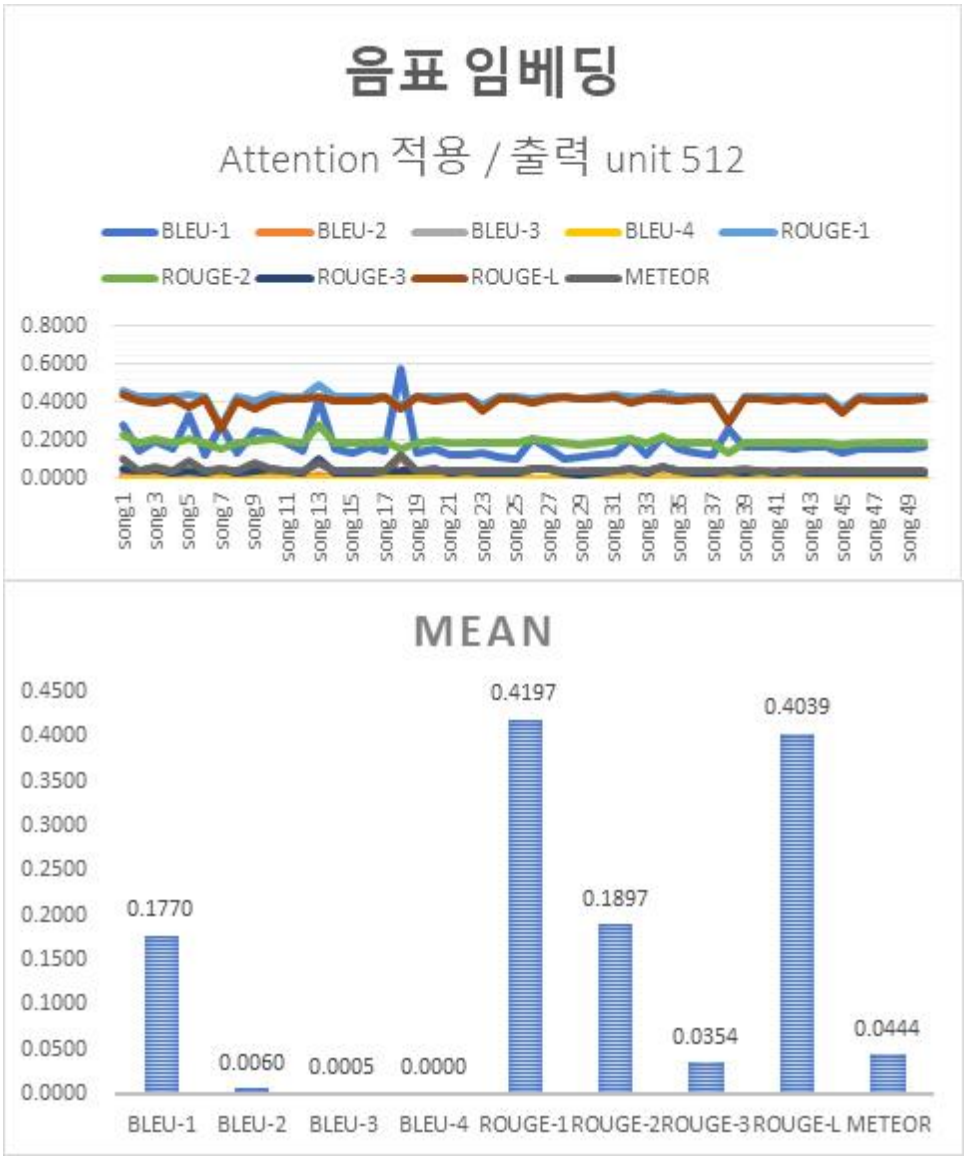
[그림 4-10] 실험 모델 별 평가 결과 자료 5



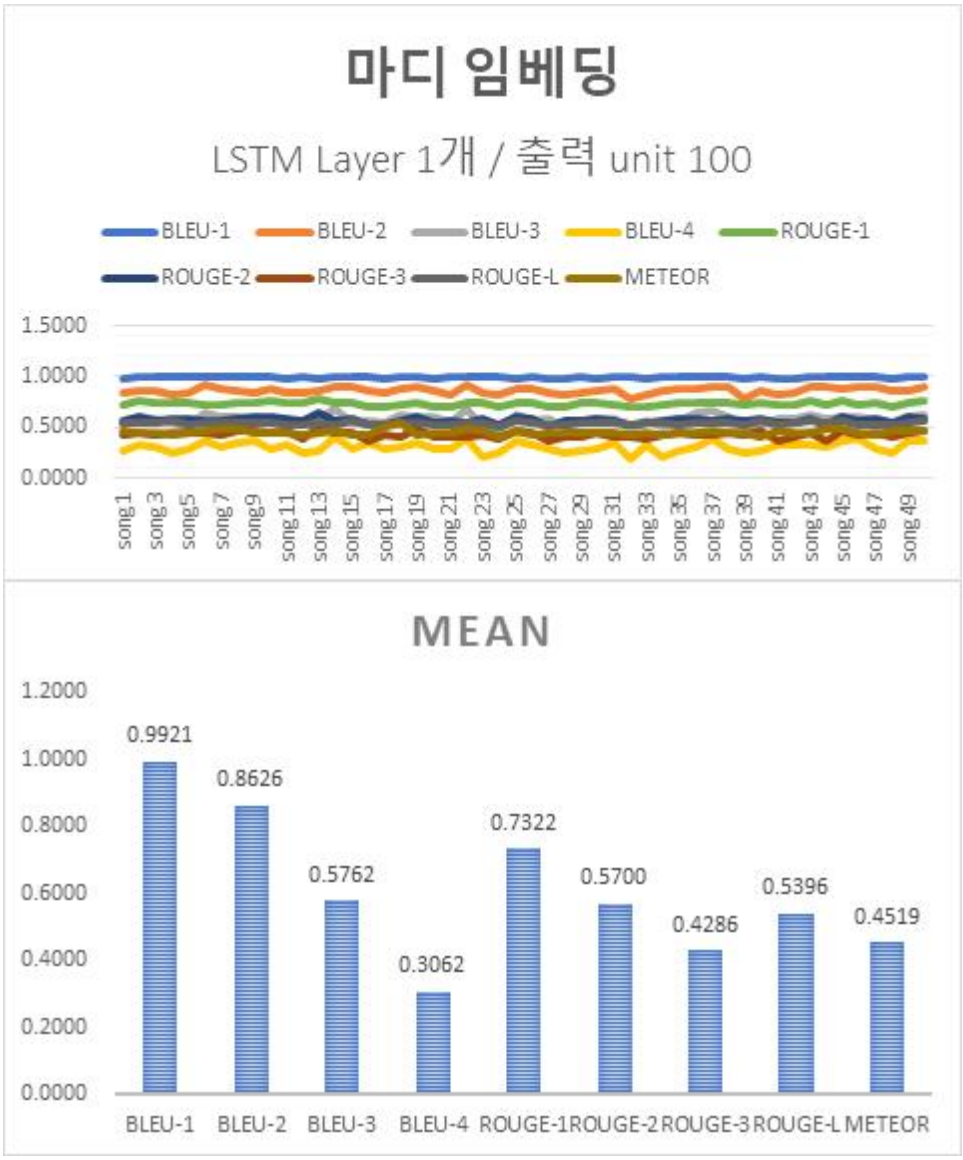
[그림 4-11] 실험 모델 별 평가 결과 자료 6



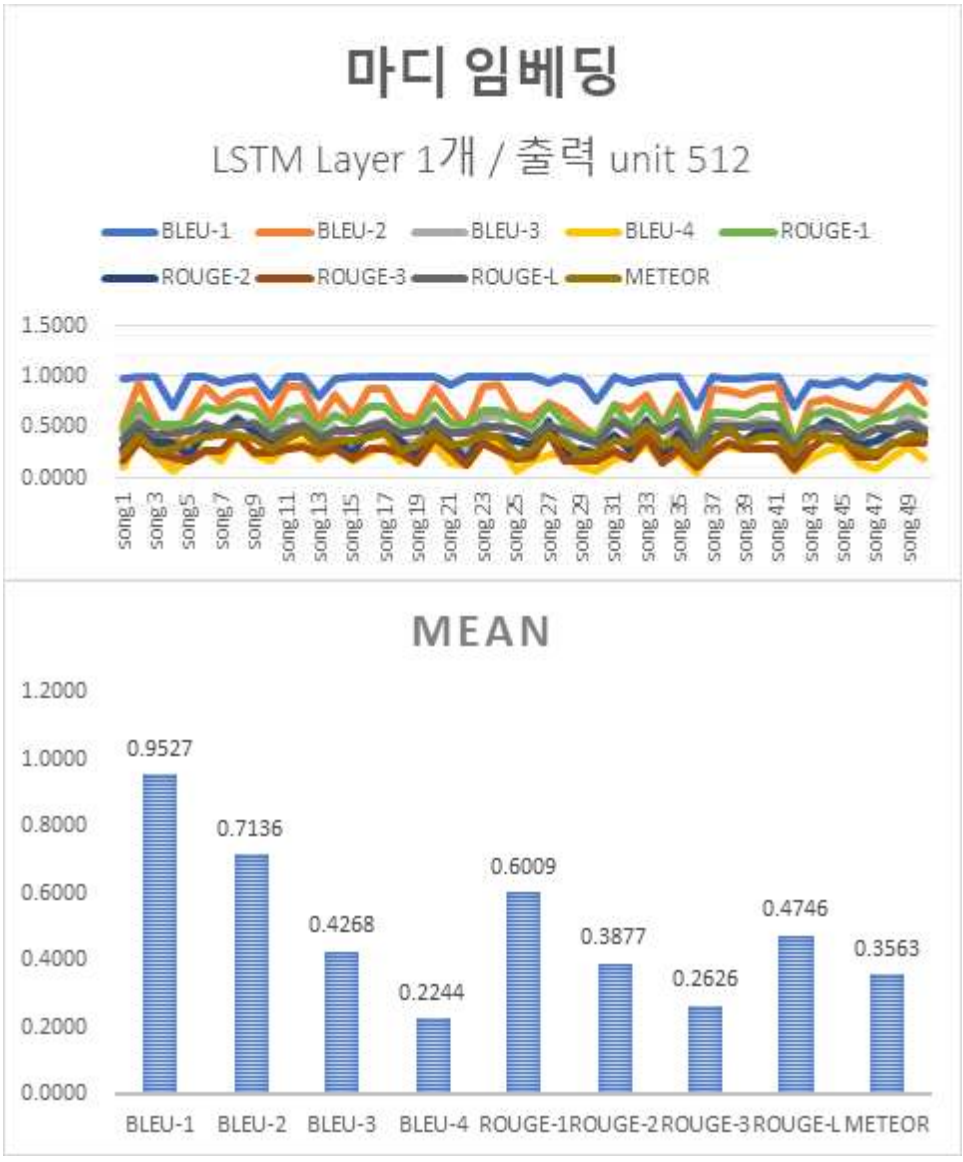
[그림 4-12] 실험 모델 별 평가 결과 자료 7



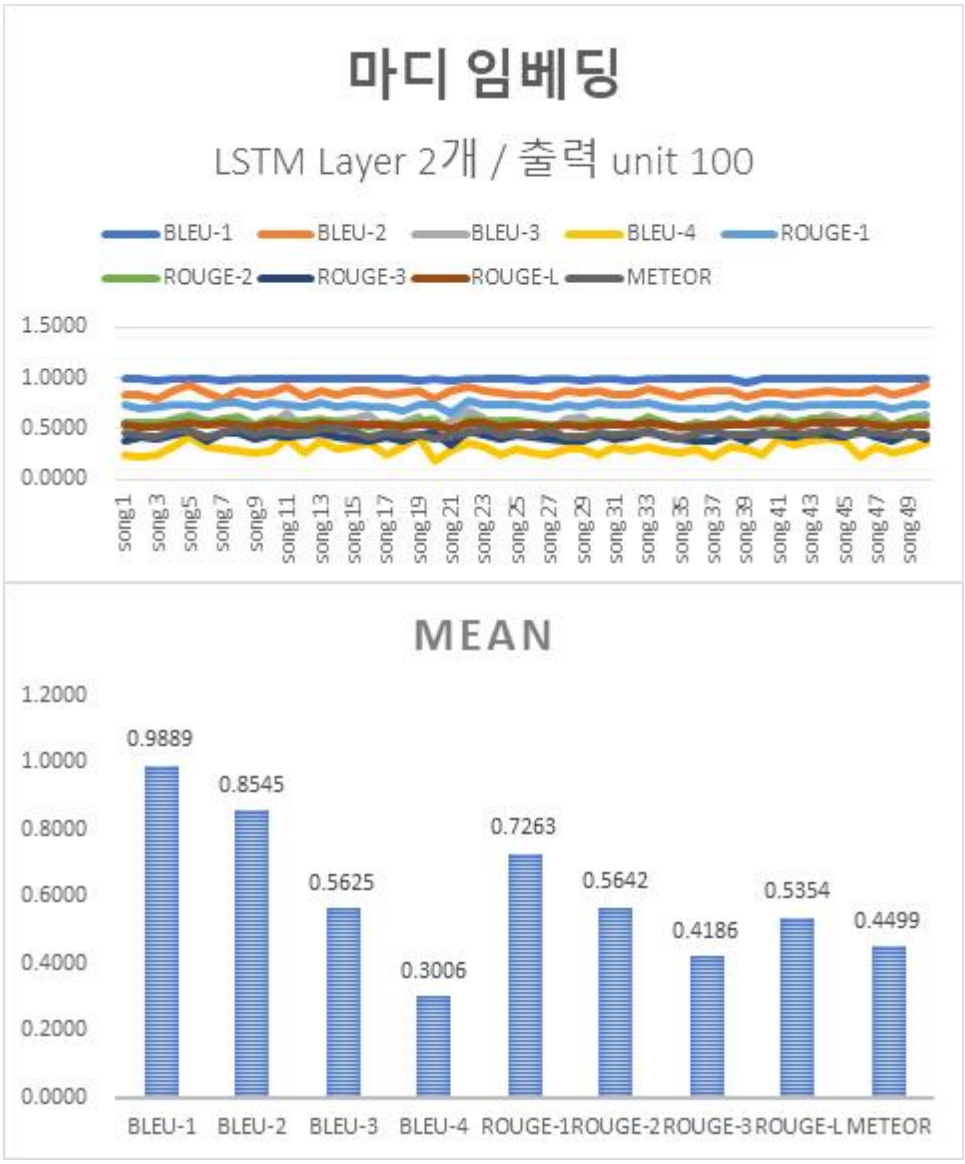
[그림 4-13] 실험 모델 별 평가 결과 자료 8



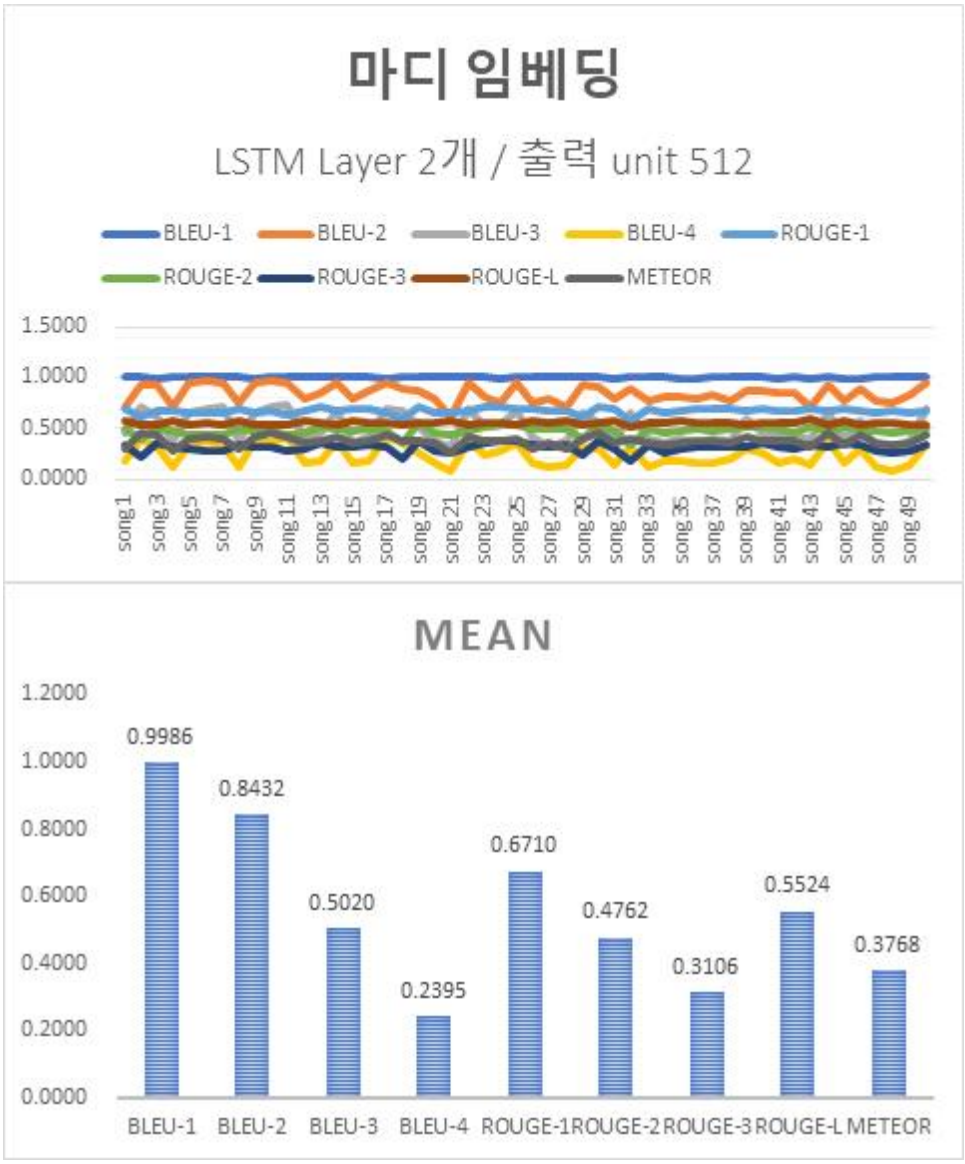
[그림 4-14] 실험 모델 별 평가 결과 자료 9



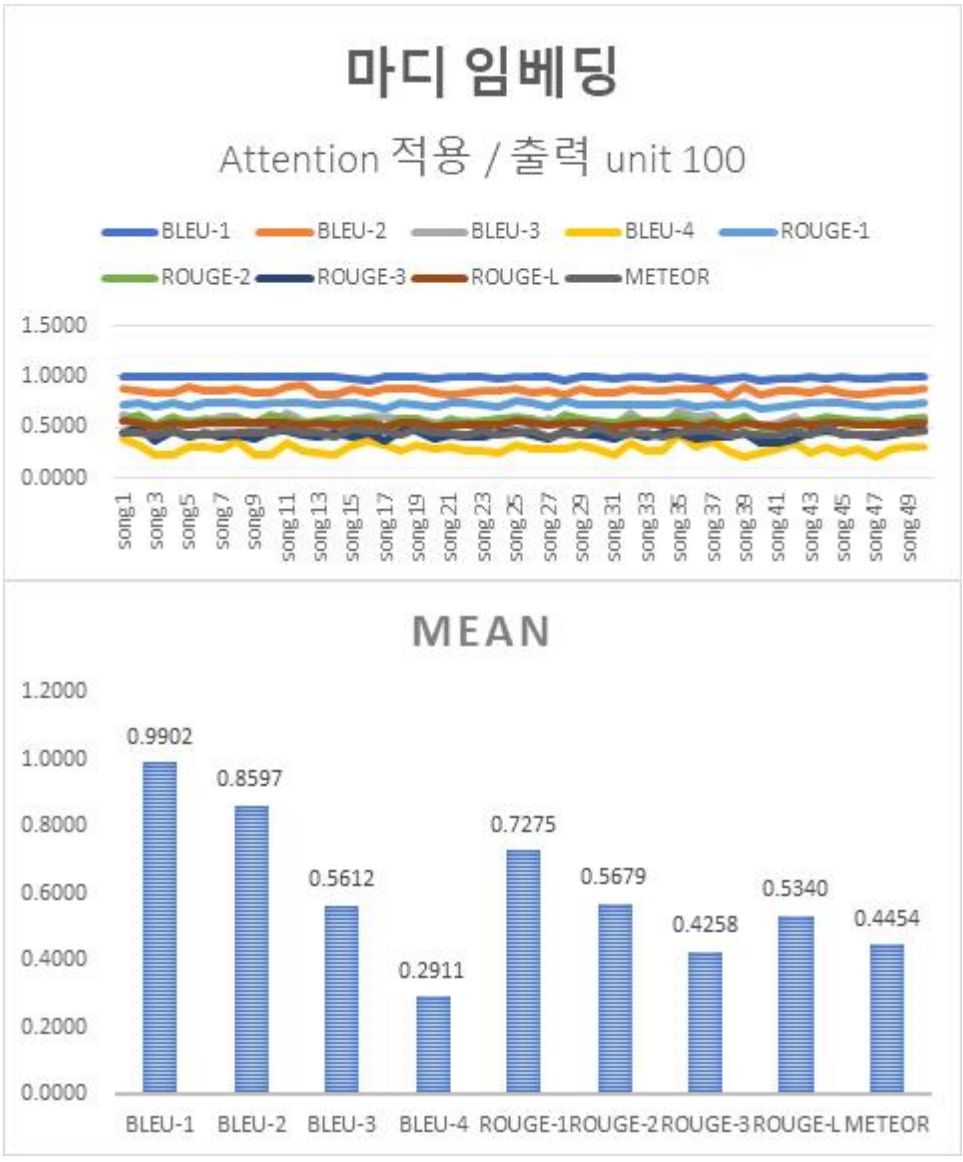
[그림 4-15] 실험 모델 별 평가 결과 자료 10



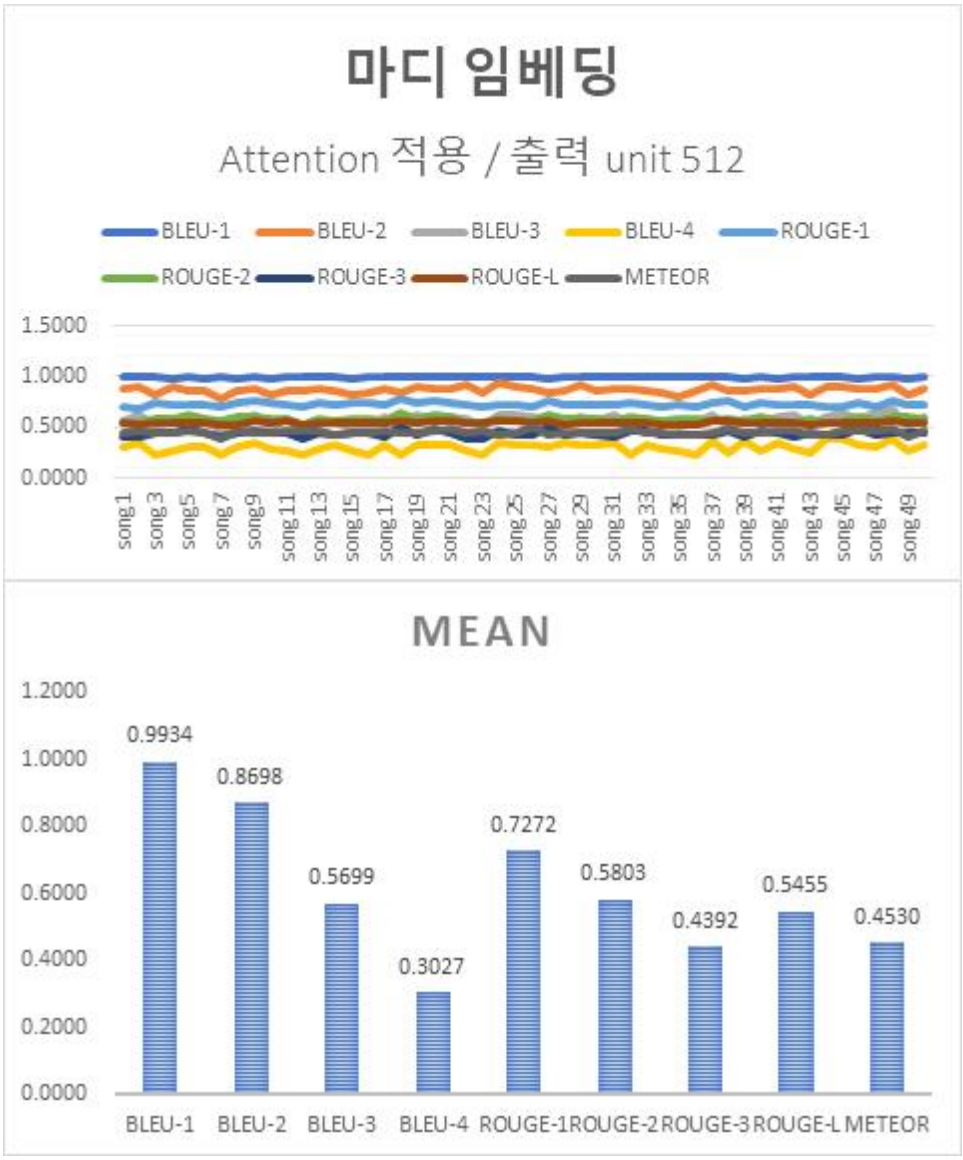
[그림 4-16] 실험 모델 별 평가 결과 자료 11



[그림 4-17] 실험 모델 별 평가 결과 자료 12



[그림 4-18] 실험 모델 별 평가 결과 자료 13



[그림 4-19] 실험 모델 별 평가 결과 자료 14

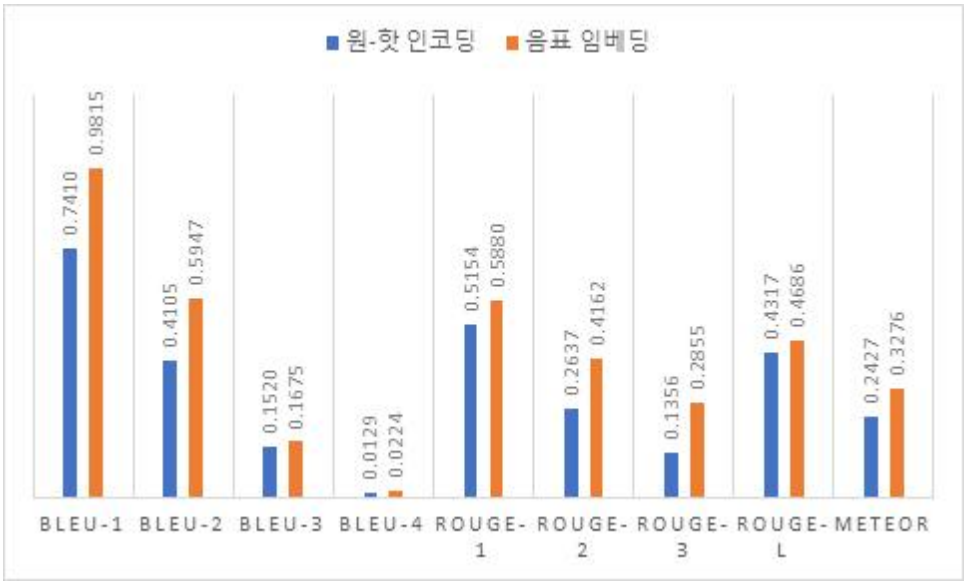
2) 원-핫 인코딩 방식과 음표 임베딩 방식의 비교 평가

4.2.1절에서 살펴본 것처럼 원-핫 인코딩 방식과 음표 임베딩 방식의 비교에서는 LSTM 층 1개를 사용한 언어 모델을 사용한다.

① 조건 : LSTM 층 출력 unit 100

		BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	METEOR
원-핫 인코딩	LSTM 1개	0.7410	0.4105	0.1520	0.0129	0.5154	0.2637	0.1356	0.4317	0.2427
음표 임베딩	LSTM 1개	0.9815	0.5947	0.1675	0.0224	0.5880	0.4162	0.2855	0.4686	0.3276

[그림 4-20] 원-핫 인코딩과 음표 임베딩 비교 평가 결과 1



[그림 4-21] 원-핫 인코딩과 음표 임베딩 비교 평가 결과 2

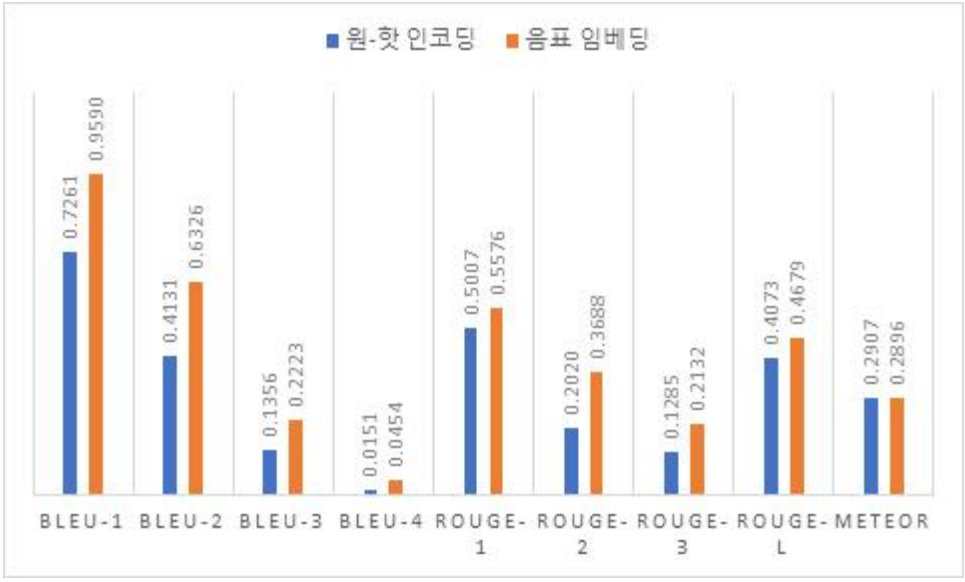
[평가 결과 분석] 첫째. 두 가지 방식 모두 n이 증가할수록 n-gram BLEU 점수와 n-gram ROUGE 점수가 낮아지는데, 그 이유는 2.2절에서 살펴본 것처럼 일반적으로 일치하는 연속된 음표의 수가 1에서 2, 3, 4로 늘어날수록 정확도(Precision)와 재현율(Recall)이 감소하므로 평가점수가 낮아지게 된다. 둘째. n-gram BLEU 점수와 n-gram ROUGE 점수의 경우 음표 임베딩 방식이 원-핫 인코딩 방식보다는 상대적으로 높은 점수를 얻었으며, 곡 전체에서 배열의 순서가 일치하는 음표의 수(LCS)로 정확도와 재현율을 구하여

F-measure 점수를 구하는 ROUGE-L 점수도 음표 임베딩 방식이 상대적으로 높은 점수를 얻었고, 음표 단위의 유사도와 곡 전체 수준에서의 유사도를 모두 고려하는 METEOR 점수도 음표 임베딩 방식이 원-핫 인코딩 방식보다는 상대적으로 높은 점수를 얻었다. 셋째. 우리가 2.3절에서 살펴본 것처럼 원-핫 인코딩의 결과 생성되는 원-핫 벡터는 희소 벡터로서 음표 간에 어떤 관계도 생성되지 않는다. 반면에 음표 임베딩 방식은 각 음표에 관계성을 부여하고 음표 사이의 연관성이나 문맥을 활용할 수 있다. 이러한 차이로 인해서 새로운 곡을 생성할 때 음표 임베딩 방식이 원-핫 인코딩 방식보다 인간이 작곡한 곡과 더 유사한 곡을 생성할 수 있다는 것을 확인할 수 있다.

② 조건 : LSTM 층 출력 unit 512

		BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	METEOR
원-핫 인코딩	LSTM 1개	0.7261	0.4131	0.1356	0.0151	0.5007	0.2020	0.1285	0.4073	0.2907
음표 임베딩	LSTM 1개	0.9590	0.6326	0.2223	0.0454	0.5576	0.3688	0.2132	0.4679	0.2896

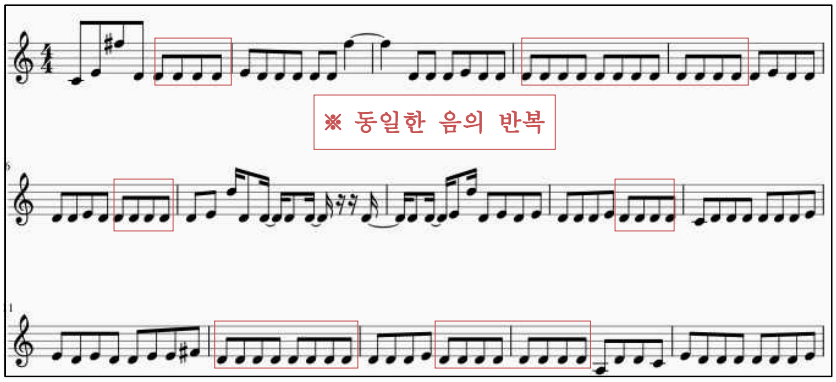
[그림 4-22] 원-핫 인코딩과 음표 임베딩 비교 평가 결과 3



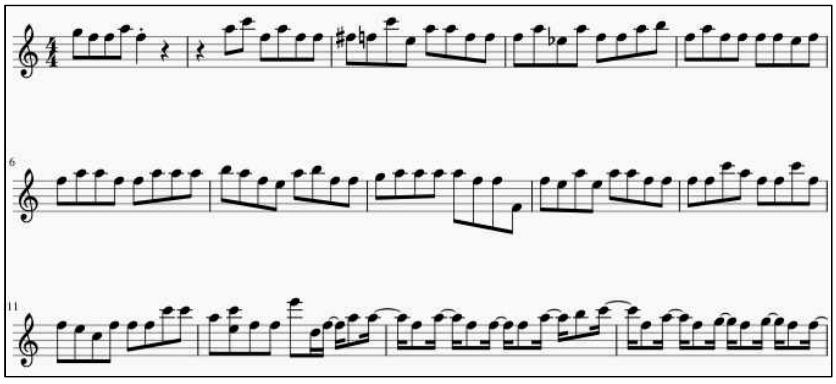
[그림 4-23] 원-핫 인코딩과 음표 임베딩 비교 평가 결과 4

[평가 결과 분석] 첫째. METEOR 점수를 제외하면 ①과 같은 결과가 나왔으며 새로운 곡을 생성할 때 음표 임베딩 방식이 원-핫 인코딩 방식보다 인간이 작곡한 곡과 더 유사한 곡을 생성할 수 있다는 것을 확인할 수 있다. 둘째. METEOR 점수의 경우 ①에 비해서 원-핫 인코딩 방식은 점수가 높아졌고 음표 임베딩 방식은 낮아졌으며 그 결과 두 방식의 점수가 거의 동일하게 나타났다. ①과 ②의 차이는 오로지 LSTM 층 출력 unit의 수이므로 출력 unit의 수가 100개에서 512개로 증가하여 가중치의 크기가 달라진 것이 결과에 영향을 주었다고 판단이 되며 이에 대한 보다 정밀한 추가 실험이 필요하다.

③ 음표 생성 결과



[그림 4-24] 원-핫 인코딩 방식의 음표 생성 결과



[그림 4-25] 음표 임베딩 방식의 음표 생성 결과 1

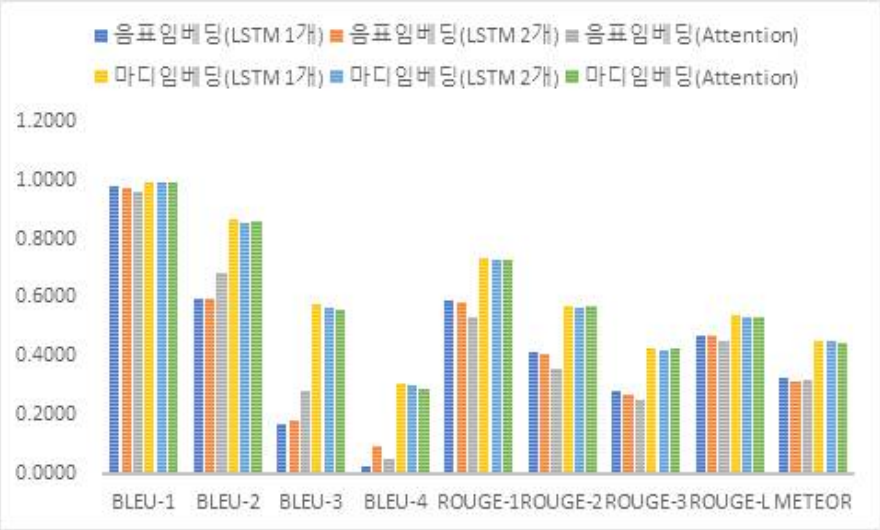
[생성 결과 분석] 첫째. 2.2절에서 살펴본 것처럼 특정 음표가 반복이 되면 될수록 평가 점수가 낮아지게 되는데 원-핫 인코딩 방식을 이용해서 생성된 곡을 보면 빨간색 네모로 표시한 부분처럼 특정 음표가 반복되어 생성되는 것을 확인 할 수 있었다. 둘째. 사람이 작곡을 할 경우 [그림 4-24]처럼 특정 음표를 과도하게 반복하지는 않을 것이고 이는 원-핫 인코딩 방식의 학습 능력이 음표 임베딩 방식보다는 떨어진다는 것을 의미한다.

3) 음표 임베딩 방식과 마디 임베딩 방식의 비교 평가

① 조건 : LSTM 층 출력 unit 100인 경우

		BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	METEOR
음표 임베딩	LSTM 1개	0.9815	0.5947	0.1675	0.0224	0.5880	0.4162	0.2855	0.4686	0.3276
	LSTM 2개	0.9750	0.5970	0.1835	0.0953	0.5845	0.4106	0.2669	0.4710	0.3133
	Attention	0.9599	0.6848	0.2832	0.0490	0.5347	0.3582	0.2515	0.4535	0.3181
마디 임베딩	LSTM 1개	0.9921	0.8626	0.5762	0.3062	0.7322	0.5700	0.4286	0.5396	0.4519
	LSTM 2개	0.9889	0.8545	0.5625	0.3006	0.7263	0.5642	0.4186	0.5354	0.4499
	Attention	0.9902	0.8597	0.5612	0.2911	0.7275	0.5679	0.4258	0.5340	0.4454

[그림 4-26] 음표 임베딩과 마디 임베딩 비교 평가 결과 1



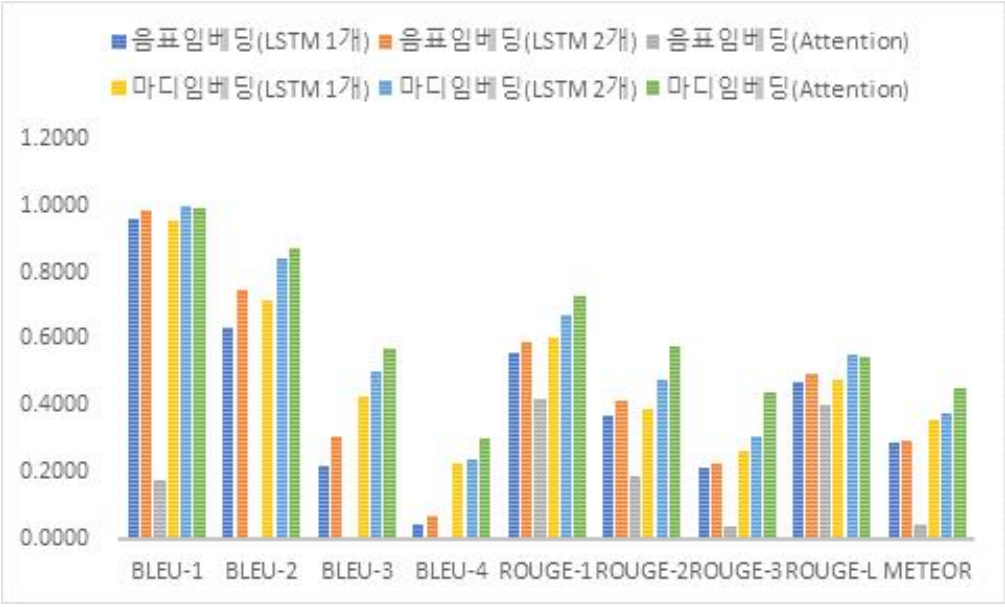
[그림 4-27] 음표 임베딩과 마디 임베딩 비교 평가 결과 2

[평가 결과] 평가 방법 전체에 걸쳐서 마디 임베딩 방식을 사용하고 LSTM 층의 수가 1개인 모델이 가장 높은 점수를 얻었다.

② 조건 : LSTM 층 출력 unit 512인 경우

		BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	METEOR
음표 임베딩	LSTM 1개	0.9590	0.6326	0.2223	0.0454	0.5576	0.3688	0.2132	0.4679	0.2896
	LSTM 2개	0.9820	0.7457	0.3052	0.0694	0.5906	0.4137	0.2278	0.4971	0.2977
	Attention	0.1770	0.0060	0.0005	0.0000	0.4197	0.1897	0.0354	0.4039	0.0444
마디 임베딩	LSTM 1개	0.9527	0.7136	0.4268	0.2244	0.6009	0.3877	0.2626	0.4746	0.3563
	LSTM 2개	0.9986	0.8432	0.5020	0.2395	0.6710	0.4762	0.3106	0.5524	0.3768
	Attention	0.9934	0.8698	0.5699	0.3027	0.7272	0.5803	0.4392	0.5455	0.4530

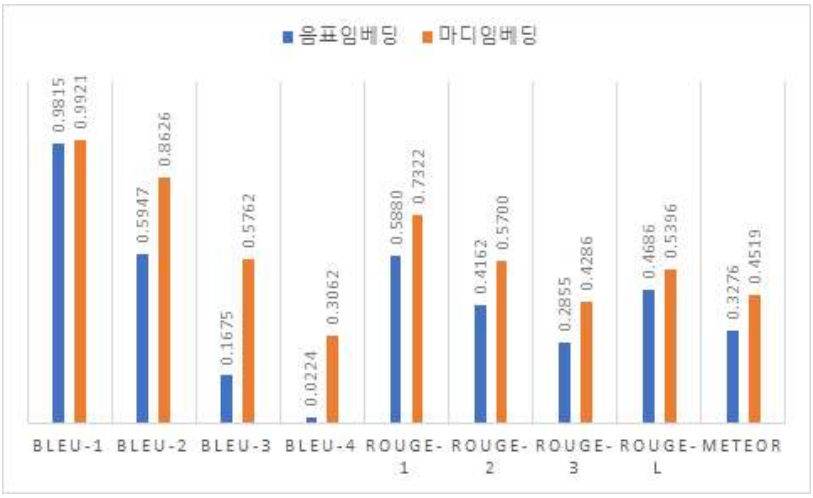
[그림 4-28] 음표 임베딩과 마디 임베딩 비교 평가 결과 3



[그림 4-29] 음표 임베딩과 마디 임베딩 비교 평가 결과 4

[평가 결과] 마디 임베딩 방식을 사용하고 어텐션 mechanism을 구현한 모델이 종합적으로 가장 높은 점수를 얻었다.

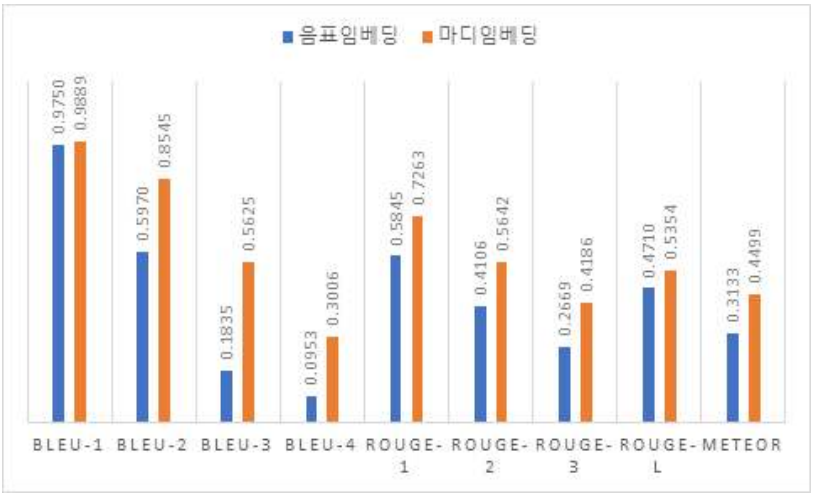
③ 조건 : LSTM 층 1개 + LSTM 층 출력 unit 100(동일한 모델 비교)



[그림 4-30] 음표 임베딩과 마디 임베딩 비교 평가 결과 5

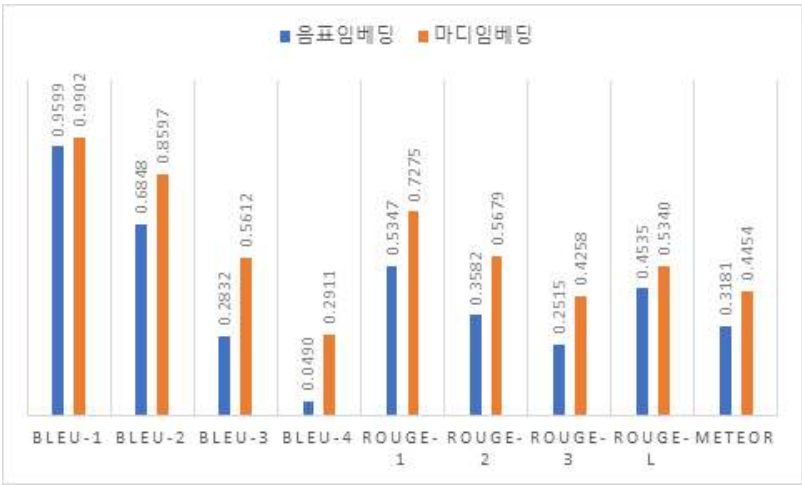
[평가 결과] 평가 방법 전체에 걸쳐서 마디 임베딩 방식이 더 높은 점수를 얻었으며, 아래 ④부터 ⑧까지의 실험 결과는 ③과 동일하다.

④ 조건 : LSTM 층 2개 + LSTM 층 출력 unit 100(동일한 모델 비교)



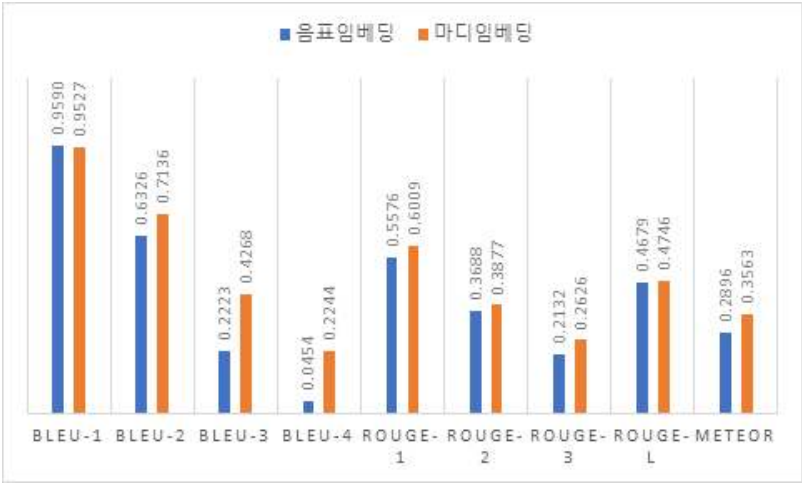
[그림 4-31] 음표 임베딩과 마디 임베딩 비교 평가 결과 6

⑤ 조건 : 어텐션 mechanism + LSTM 층 출력 unit 100(동일한 모델 비교)



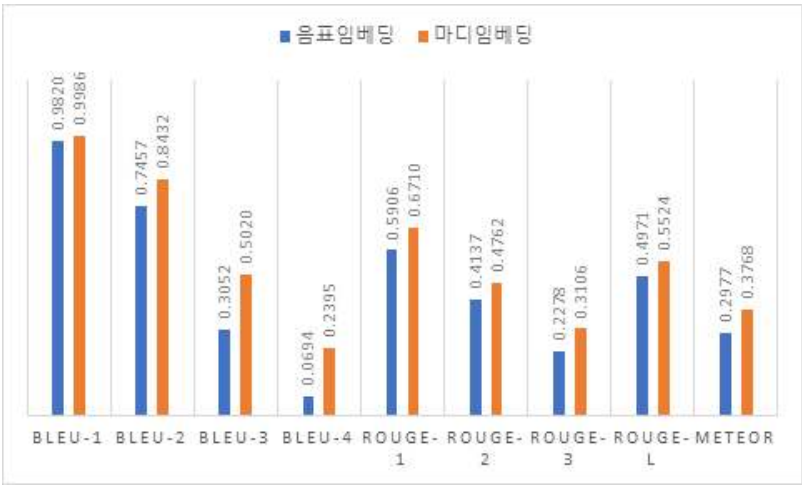
[그림 4-32] 음표 임베딩과 마디 임베딩 비교 평가 결과 7

⑥ 조건 : LSTM 층 1개 + LSTM 층 출력 unit 512(동일한 모델 비교)



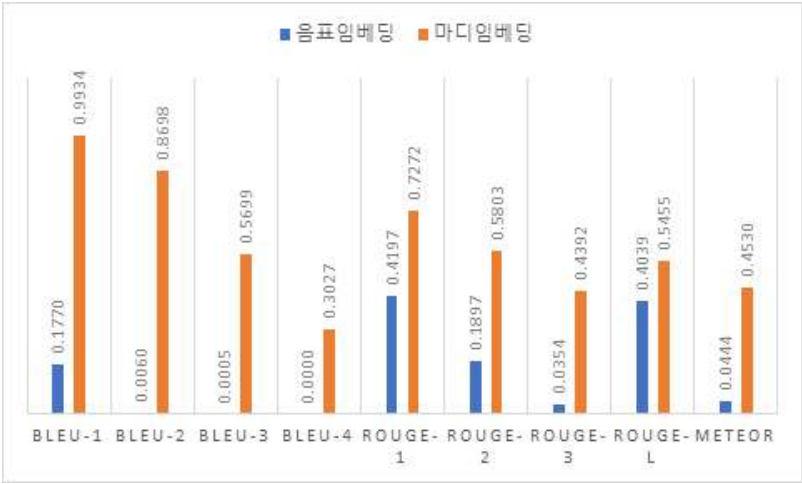
[그림 4-33] 음표 임베딩과 마디 임베딩 비교 평가 결과 8

⑦ 조건 : LSTM 층 2개 + LSTM 층 출력 unit 512(동일한 모델 비교)



[그림 4-34] 음표 임베딩과 마디 임베딩 비교 평가 결과 9

⑧ 조건 : 어텐션 mechanism + LSTM 층 출력 unit 512(동일한 모델 비교)

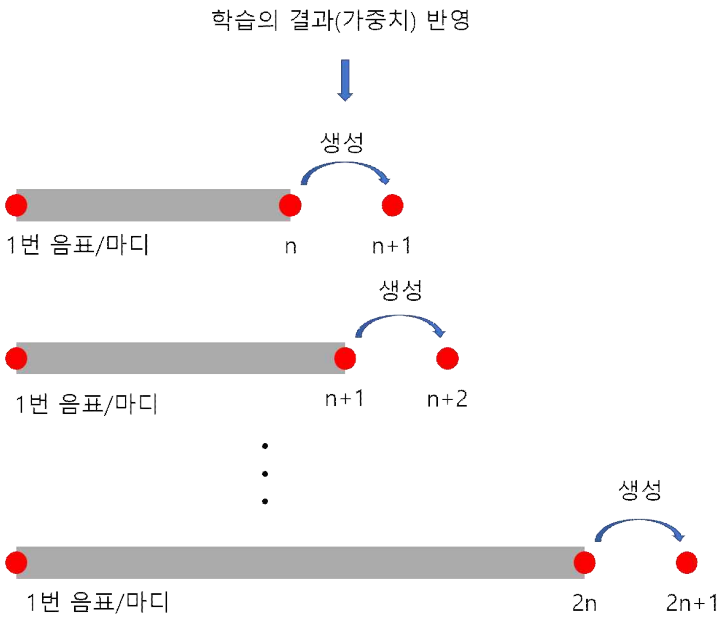


[그림 4-35] 음표 임베딩과 마디 임베딩 비교 평가 결과 10

⑨ 음표 임베딩 방식과 마디 임베딩 방식의 평가 결과 분석

동일한 조건을 가진 동일한 모델에서는 마디 임베딩 방식이 음표 임베딩 방

식보다 더 높은 평가 점수를 얻어서 인간이 작곡한 곡과 유사한 곡을 생성한다는 것을 확인할 수 있었다. 그 이유는 다음과 같다. 2.3절에서 살펴본 것처럼 임베딩은 음표나 마디와 같은 정보를 벡터의 형태로 숫자화 하되 원-핫 인코딩과는 달리 각각의 음표나 마디에 관계성을 부여해서 LSTM 층이 음표와 음표 또는 마디와 마디 사이의 관계를 학습하는 것이 가능하게 함으로써 자동 작곡 모델이 새로운 음표나 마디를 생성할 때 무작위로 생성하는 것이 아니라 아래 그림처럼 학습의 결과를 바탕으로 앞에 있는 음표 또는 마디와 가장 잘 어울리는 음표나 마디를 생성하게 된다.

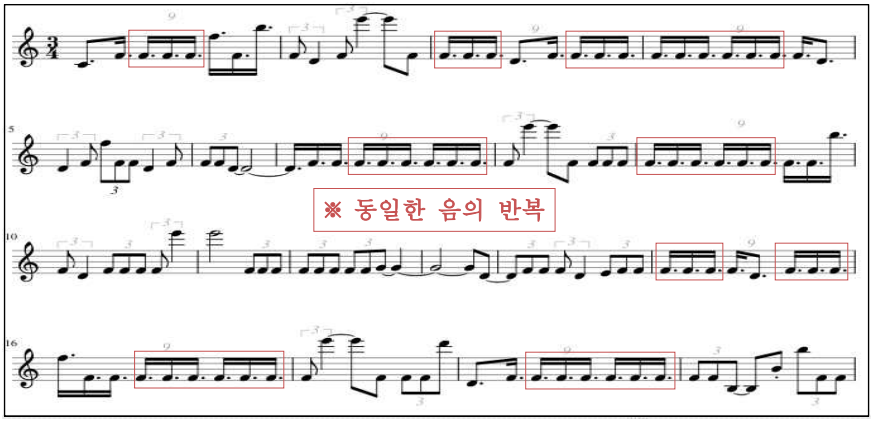


[그림 4-36] 새로운 음표/마디 생성

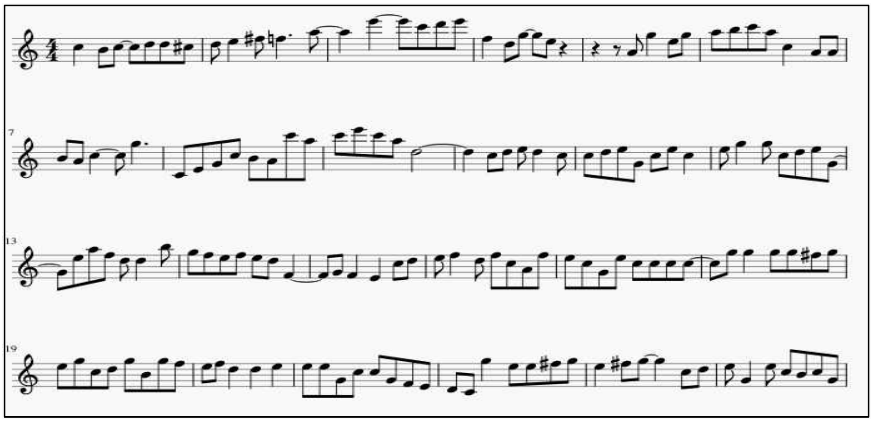
임베딩의 효과는 결국 학습 데이터 사이의 관계를 얼마나 다양하게 표현하고 학습할 수 있는지에 따라 달라지는데, 음표 임베딩과 마디 임베딩은 둘 다 임베딩이라는 공통된 방식을 사용하지만 임베딩의 단위(unit)가 음표와 마디로 다르기 때문에 그 효과가 동일하지 않다. 우리가 학습에 사용한 노래는 828곡이며, 음표의 총 수는 156,405개이고, 음표의 종류는 384개, 마디의 종류는 9,149개이다. 384개보다는 9,149개로 만들 수 있는 관계의 다양성이 더 높기

때문에 마디 임베딩 방식이 음표 임베딩 방식보다 학습의 효과가 더 크다. 이런 차이로 인해서 음표 임베딩보다는 마디 임베딩이 인간이 작곡한 곡과 더 유사한 곡을 생성한다.

⑩ 음표 생성 결과



[그림 4-37] 음표 임베딩 방식의 음표 생성 결과 2



[그림 4-38] 마디 임베딩 방식의 음표 생성 결과

[생성 결과 분석] 앞에서 살펴 본 원-핫 인코딩 방식보다는 적지만 음표 임베딩 방식의 경우도 [그림 4-37]처럼 동일한 음을 반복해서 생성하는 것을 확인 할 수 있었다. ⑨에서 살펴본 것처럼 마디 임베딩 방식이 음표 임베딩 방식보다는 인간이 작곡한 곡과 더 유사하게 곡을 생성한다.

V. 결 론

본 연구의 목적은 첫째. 딥러닝 기반 자동 작곡에서 생성 곡의 정량적 평가 방법을 연구하기 위해서 다른 분야의 정량적 평가 방법을 적용함에 있어서의 가능성과 한계점을 확인하는 것이고, 둘째. 기존의 음표 처리 방식인 원-핫 인코딩 방식과 음표 임베딩을 적용한 방식의 품질의 차이를 평가하는 것이고, 셋째. 자연어 처리 분야에서 문자 임베딩과 단어 임베딩이 가능하듯이 음악에는 음표와 마디가 있으므로 임베딩 적용 단위(unit)를 세분화하여 음표 임베딩 외에 새로이 마디 임베딩을 구현하여 두 방식의 품질 차이를 평가하는 것이다.

첫째, 우리는 머신 러닝의 다른 분야에서 사용되는 정량적 평가 방법을 자동 작곡 분야에 적용함에 있어서, 이미지 캡셔닝 모델의 평가에 사용되는 CIDEr은 적용하기 어려우나, 자연어 처리 분야에서 사용되는 정량적 평가 방법인 BLEU, ROUGE, METEOR의 경우에는 적용이 가능하다는 것을 확인하였다.

둘째, 원-핫 인코딩 방식과 음표 임베딩 방식을 정량적 평가 방법을 이용하여 비교한 결과 원-핫 인코딩 방식 보다는 음표 임베딩 방식이 인간이 작곡한 곡과 더 유사하게 곡을 생성한다는 것을 확인하였다.

셋째, 음표 임베딩 방식과 마디 임베딩 방식을 비교 평가한 결과 음표 임베딩 방식보다는 마디 임베딩 방식이 인간이 작곡한 곡과 더 유사하게 곡을 생성한다는 것을 확인하였고, 자동 작곡에서 마디 임베딩 방식을 사용하기를 제안하였다.

우리는 실험을 통해서 본 연구의 세 가지 목적을 달성할 수 있었다. 그러나 자연어 처리 분야에서 사용되는 정량적 평가 방법이 음악적 특성을 온전히 반영하는 것은 아니다. 향후에는 자동작곡에 적합한 새로운 평가 방법을 연구하고자 한다.

참 고 문 헌

1. 국내문헌

김양훈, 황용근, 강태관. & 정교민. (2016). LSTM 언어모델 기반 한국어 문장 생성. The Journal of Korean Institute of Communications and Information Sciences '16-05 Vol.41 No.05

안성만, 정여진, 이재준. & 양지헌. (2017). 한국어 음소 단위 LSTM 언어모델을 이용한 문장 생성. J Intell Inform Syst 2017 June: 23(2): 71~88

2. 국외문헌

Yang. L. C. & Lerch. A. (2018). On the evaluation of generative models in music

Logan. B., Ellis. D. P.W, & Berenzweig. A. (2003). Toward Evaluation Techniques for Music Similarity

Papineni. K., Roukos. S., Ward. T. & Zhu. W. J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation

Lin. C. Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries

Banerjee. S. & Lavie. A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments

Hochreiter. S. & Schmidhuber. J. (1997). LONG SHORT-TERM MEMORY Neural Computation 9(8):1735 – 1780, 1997

Mikolov. T., Karafiat. M., Burget. L., Cernock. J. & Khudanpur. S. (2010). Recurrent neural network based language model

Sak. H., Senior. A. & Beaufays. F. (2014). Long Short-Term Memory

Recurrent Neural Network Architectures for Large Scale Acoustic Modeling

Sutskever. I., Vinyals. O. & Le. Q. V. (2014). Sequence to Sequence Learning with Neural Networks

Jozefowicz. R., Zaremba. W. & Sutskever. B. (2015). An Empirical Exploration of Recurrent Network Architectures

Chu. H., Urtasun. R. & Fidler. S. (2016). SONG FROM PI: A MUSICALLY PLAUSIBLE NETWORK FOR POP MUSIC GENERATION

Bretan. M., Weinberg. G. & Heck. L. (2016). A Unit Selection Methodology for Music Generation Using Deep Neural Networks

Wang. Z., Wang. D., Zhang. Y. & Xia. G. (2020). LEARNING INTERPRETABLE REPRESENTATION FOR CONTROLLABLE POLYPHONIC MUSIC GENERATION

Wang. Z., Zhang. Y., Zhang. Y., Jiang. J., Yang. R., Zhao. J. & Xia. G. (2020). PIANOTREE VAE: STRUCTURED REPRESENTATION LEARNING FOR POLYPHONIC MUSIC

Vedantam. R., Zitnick. C. L. & Parikh. D. (2015). CIDEr: Consensus-based Image Description Evaluation

Mikolov. T., Chen. K., Corrado. G. & Dean. J. (2013). Efficient Estimation of Word Representations in Vector Space

Pennington. J., Socher. R. & Manning. C. D. (2014). GloVe: Global Vectors for Word Representation

Alex G. (2008). Supervised Sequence Labelling with Recurrent Neural Networks

Bahdanau. D., Cho. K. H. & Bengio. Y. (2014). NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Xu. K., Ba. J. L., Kiros. R., Cho. K. H., Courville. A., Salakhutdinov. R., ... Bengio. Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual 어텐션

Yang. Z., Yang. D., Dyer. C., He. X., Smola. A. & Hovy. E. (2016). Hierarchical 어텐션 Networks for Document Classification

Lin. Z., Feng. M., Santos. S. N., Yu. M., Xiangl B., Zhou. B. & Bengio. Y. (2017). A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING

"Understanding LSTM Network", colah's blog, Posted on August 27, 2015, accessed Nov.16, 2020, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

ABSTRACT

Study on the Method of Creating and Quantitatively Evaluating Songs using bar Embedding in Automatic Composition Based on Deep Learning

Lee, Young-Bae

Major in Futures Convergence Strategy Consulting

Dept. of Futures Convergence Consulting

Graduate School of Knowledge Service Consulting

Hansung University

Research on generating music using deep learning has been conducted in various ways, but research on evaluation methods for the generated music is relatively small. Evaluation is very important because it is difficult to improve the performance of the automatic composition model in a state where the quality of the generated song cannot be evaluated, and criteria are also important for evaluation. Since the most complete music so far is a human-composed song, we set a human-composed song as a criterion for evaluation and quantitatively evaluate how similar the automatic composition model creates songs to human-composed songs. A quantitative evaluation method with an evaluation criteria that could result in a higher evaluation score as the song was created, similar to a song composed by humans was needed. Since the quantitative evaluation methods studied by previous researchers are not abundant, this study analyzed in detail the quantitative evaluation methods used in other fields of deep learning such as machine translation

and image captioning, and then I applied them to deep learning-based automatic composition models and evaluated them.

Since all automatic composition models are mathematical models, the notes must be converted into numerical data that the model can recognize in order to implement the model. The method of converting notes into numerical data includes one-hot encoding method using only two integers of 0 and 1, and note embedding method implemented by applying word embedding, a method that converts words in the field of natural language processing. The automatic composition model was implemented using both methods, and as described above, using the quantitative evaluation method used in other fields of deep learning, it was evaluated which method has created songs more similar to the songs composed by humans. In addition, just as character embedding and word embedding are possible in the natural language processing, music has notes and bars, so the unit to which embedding is applied has been subdivided to implement a new bar embedding method. It was evaluated quantitatively which method has created songs more similar to the human-composed songs.

Through this, it was confirmed that the quantitative evaluation methods in other fields of deep learning can and cannot be applied to the field of automatic composition. Among the one-hot encoding method and note embedding method, the score of the note embedding method is higher. And among the newly implemented bar embedding method and note embedding method, the evaluation score of the bar embedding method was higher. Through the results of this study, it was confirmed that the bar embedding method creates songs more similar to human-composed songs than the conventional one-hot encoding method or note embedding method, so we propose to use the bar embedding method in automatic composition.

【Keywords】 automatic composition, quantitative evaluation, one-hot encoding, note embedding, bar embedding