Deep Learning for AI Commenter

* NUGU Chess-

Team 5

김도혁, 김재창, 명성현, 성해빈, 유세훈

Index

1. 개요(Introduction)
2. 관련 연구(Related Works)
3. 주요 기법(Main model or Methods)
   1. Word2Vec
   2. TF-IDF
   3. DNN
   4. CNN
4. 실험(Experiments)
   1. 데이터셋(Dataset)
   2. 실험결과(Results)
5. 결론(Conclusion)
6. 참고자료(References

**1. 개요(Introduction)**

Team5는 음성인식기술을 이용한 AI스피커 NUGU candle에 맞는 play를 구현하는 과제에서 체스과 관련된 **NUGU체스**를 구현하였다. 특히, NUGU체스는 기존의 승리를 위한 체스AI와 차별화를 위해, 각 수를 해설해주는 AI를 딥러닝을 통해 학습하였다. 본 보고서는 관련 기술에 초점을 맞추었다.

**2. 관련 연구(Related Works)**

체스 AI와 관련된 연구는 주로 직접 기보를 분석해 관련된 특징을 뽑아내거나(feature selection) 체스판의 현재 상황을 적당한 함수로 직접 근사한 방법이 대부분이다. Deep Blue[1]는 1997년 당시 세계 체스 챔피언인 Garry Kasparov를 꺾었던 체스 모델로, 체스판을 평가하는 복잡한 함수를 설계하고 게임 트리를 가지치기(pruning)와 함께 병렬적으로 탐색한 방법을 사용했다.

최근에는 딥러닝을 이용한 체스 AI가 제시되기도 하였다. DeepChess[2]는 CNN으로 기보를 통해 평가 함수를 학습하였고 기존 체스 엔진인 FALCON이나 CRAFTY에 승리하기도 했다. 이밖에도 체스와 관련된 많은 모델이 제시되었으나 모두 승률을 계산하거나 다음 수를 예측하는 등의 승/패 중심의 모델이다.

**3. 주요 기법(Main model or Methods)  
3.1 Word2Vec**

문장에 있는 단어를 컴퓨터가 처리할 수 있게 변환할 때 ‘one-hot encoding’ 방식을 많이 사용했다. ‘one-hot encoding’은 단어의 수만큼의 0과 1로 이루어진 벡터를 정의하여 문장에 해당 단어가 있으면 1, 없으면 0으로 표현하는 방식이다. 그러나 ‘one-hot encoding’ 방식은 직관적이지만 단어의 수가 증가할 때 벡터의 공간이 지수적으로 크게 증가한다는 단점이 있다.

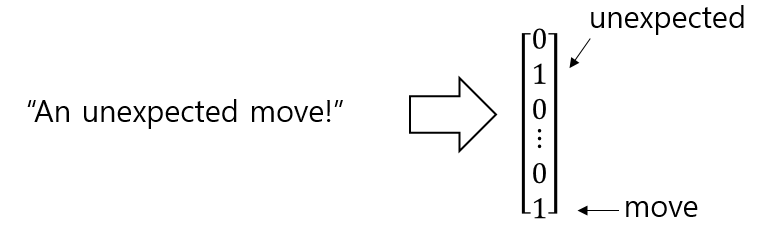


Fig. 1. ‘one-hot encoding’의 예시.

Word Embedding은 단어의 수보다 낮은 차원으로 근사하는 기법으로 ‘one-hot encoding’의 단점을 극복하기 위해 제안되었다. Word2Vec[3]은 인공 신경망(Neural Networks) 기반의 word embedding 모델 중 널리 쓰이는 모델로 크게 CBOW(Continuous Bag-of-Words)와 Skip-gram의 두가지 방식으로 나뉜다.

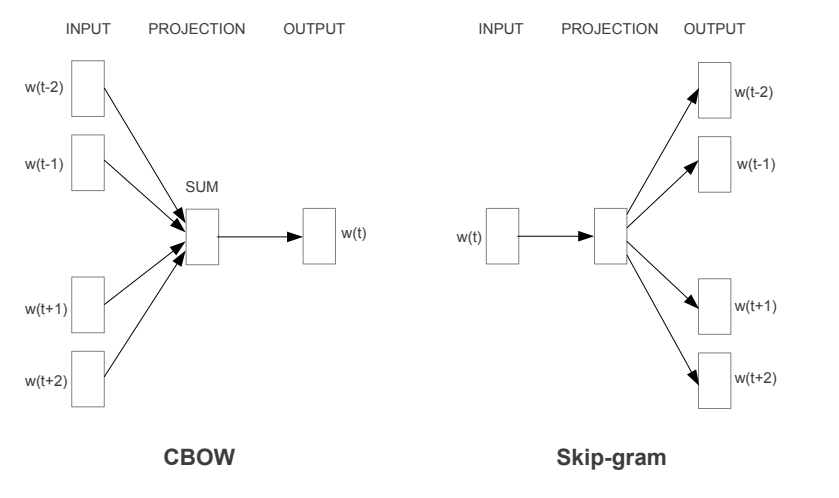


Fig. 2. CBOW vs. Skip-gram

CBOW는 문맥으로부터 특정 단어를 예측하며, Skip-gram은 특정 단어에 대해 문맥을 추론하는 방식으로 학습된다. 예를 들어 “나는 버거킹을 정말 사랑한다.” 라는 문장에서 “정말”이라는 단어에 대해 CBOW는 “나는 버거킹을 { } 사랑한다.”에서 빈칸을 학습하려 하고, Skip-gram은 “나는 { } 정말 { }”에서 빈칸을 학습하는 방식이다.

**3.2 TF-IDF**

특정 문장에 속한 단어들은 학습된 Word2Vec을 통해 벡터로 변환될 수 있다. 어떤 기보에 대한 주석을 학습하기 위해 이 벡터들의 평균값이 사용될 수 있다. 그러나 단순한 평균값보다는 단어의 중요도에 따라 가중치를 둬서 평균을 구하는 것이 더 바람직하다.

TF-IDF(Term Frequency – Inverse Document Frequency)는 어떤 단어가 특정 문서 내에서 얼마나 중요한지 나타내주는 지표이다. 특정 문서 와 단어 에 대해 TF-IDF는 다음과 같이 계산된다.

여기서 는 문서 내에 단어 가 몇 번 등장 했는지를 뜻한다. 는 모든 문서에 대해 단어 가 들어있는 문서 수에 반비례하며 다음과 같다.

**3.3 DNN**

DNN(Deep Neural Network)은 입력층과 출력층 사이에 은닉층(Hidden layers)이 2개 이상 들어있는 인공 신경망 모델을 통칭한다. 은닉층이 여러 개 들어있기 때문에 DNN은 입력 값에 대한 비선형 데이터들을 만들어낼 수 있다. 본 프로젝트에서 사용한 DNN은 Fully-connected layer들로만 이루어진 가장 보편적인 형태의 인공신경망이다. 필요에 따라 Fully-connected 이외에도 다양한 layer를 추가할 수 있다.

**3.4 CNN**

CNN(Convolutional Neural Network)는 크게 Convolutional Layer들과 Fully-connected Layer들의 두 부분으로 구성된다. 먼저 첫번째 부분인 Convolutional Layer는 고정된 Convolution 필터를 Input feature map 상에서 이동시키면서 다음 단계의 Hidden layer를 만든다. 따라서 한 Layer에 필요한 parameter의 개수는 fully connected layer보다 현저히 적다. 각 Convolutional Layer 사이에는 공간의 크기를 줄이기 위한 Pooling 레이어가 추가될 수 있다. 이렇게 필요한 만큼의 Convolutional Layer를 거친 데이터는 Raw input data보다 크기가 훨씬 줄어들었기 때문에 Fully-connected Layer를 통해 더 정교한 관계식을 만들 수 있다.

**4. 실험(Experiments)**

**4.1 데이터셋(Dataset)**

실험을 위한 데이터는 gameknot.com, gambitchess.com, angelfire.com 등의 온라인 체스 기보 사이트에서 추출했다. 총 12313판의 게임 데이터를 얻었고, 전처리를 거쳐 293070개의 데이터를 Training/Testing 용도로 사용할 수 있었다.

하나의 데이터는 현재 게임판 상태, 이번 턴에 말이 어떻게 움직였는지, 그리고 이 수에 대한 전문가의 영문 코멘트로 이루어져 있다. 게임판 상태는 Forsyth-Edwards 표기법(FEN)으로 변환했다. 하나의 FEN은 6가지 정보를 저장한다.

- 기물의 위치

- 다음의 둘 차례(흑 또는 백)

- 각 참가자가 캐슬링이 가능한지

- 앙파상을 당할 수 있는 폰이 있는지

- 폰의 전진이나 기물 잡기 없이 몇 수가 지났는지

- 전체 몇 수가 지난 상태인지

이번 턴에 둔 수는 말의 종류와 목적지, 그리고 적군의 말을 잡았는지를 기록하는 Standard algebraic notation 형태로 작성했다. 결과적으로 FEN과 SAN 표현을 합치면 각 말의 위치와 마지막 움직임을 행렬로 표현할 수 있다.

Word2vec은 20번 이하로 등장하는 단어를 제거하고, 주위 10개 단어를 고려(window size: 10)하게 한 결과 5016개의 50차원 벡터가 만들어졌다.

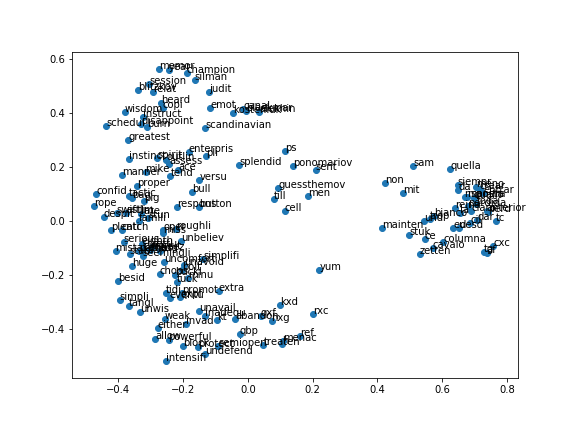
****

Fig. 3. 무작위로 뽑은 150개 단어들을 시각화한 그림

**4.2 실험결과(Results)**

|  |  |  |  |
| --- | --- | --- | --- |
| Model | DNN(3-layers) | CNN1(1CV+2FC) | CNN2(2CV+2FC) |
| RMSE | 0.062575 | **0.061483** | 0.061527 |
| # of parameters | 334770 | 222418 | 198260 |

Table. 1. 각 모델의 RMSE. DNN은 3개의 은닉층(256, 512, 128)이고, CNN1은 32 channel convolutional layer +(256,128) fully connected layer이며, CNN2는 (32, 64) channel의 2 convolutional layer+(512,256) fully connected layer이다.

구현은 pytorch를 이용하였으며, 데이터를 train set: 211400, test set: 81670으로 나누었다. DNN과 CNN을 비교하기 위해 은닉층의 노드 개수를 hyperparameter로 보고 5-fold cross validation와 grid search로 결정하였고, test set의 RMSE는 표와 같다. 두 모델 다 adam optimizer와 early stopping 기법을 이용하였고 batch size는 5000, learning rate를 0.001로 통일하였다. DNN의 최적 노드 개수는 (256, 512, 128)이고, CNN1은 32 channel convolutional layer +(256,128) fully connected layer, CNN2는 (32, 64) channel의 2 convolutional layer+(512,256) fully connected layer 이다.

**5. 결론(Conclusion)**

이번 과제에서 체스 기보에 대한 주석이 있는 데이터셋을 바탕으로 해설AI를 학습했다. 모델은 DNN과 CNN을 이용했으며, RMSE 값을 이용해 더 좋은 모델을 평가하였다.

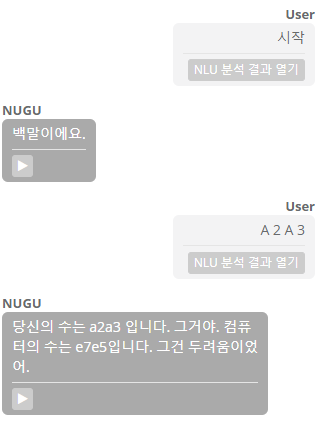
그 결과 ~이 뛰어났다. 그리고 학습된 모델을 이용해 구현한 ~는 다음과 같다. => 그림 몇 개 붙이고 끝?

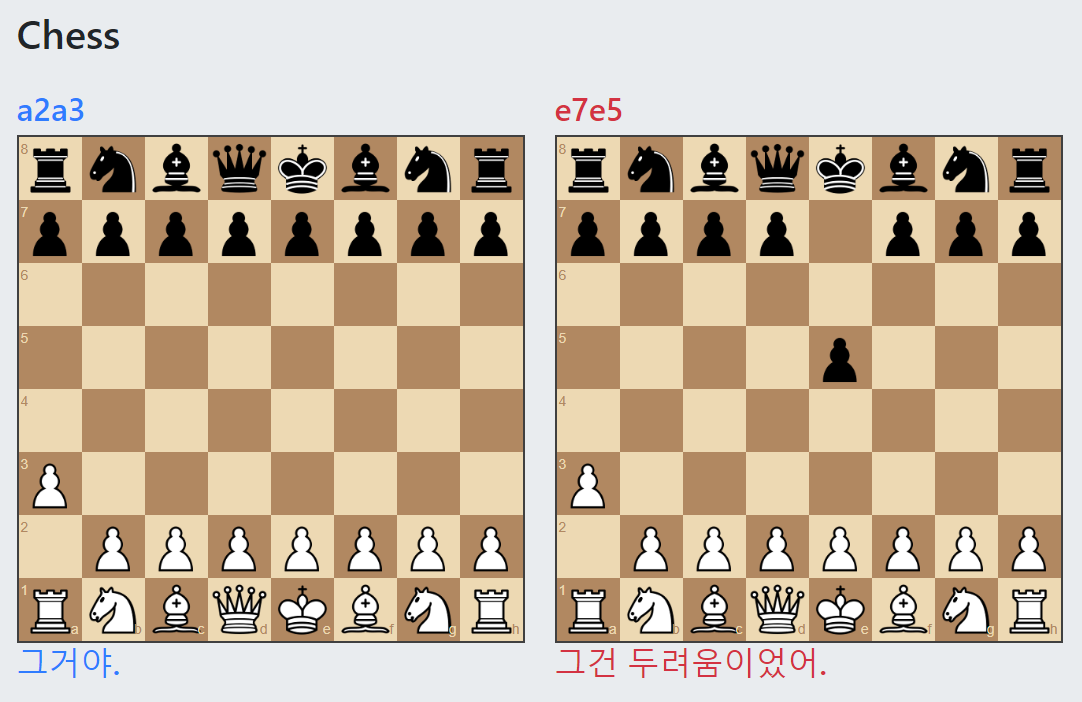
**6. Play 구조(Play Structure)**

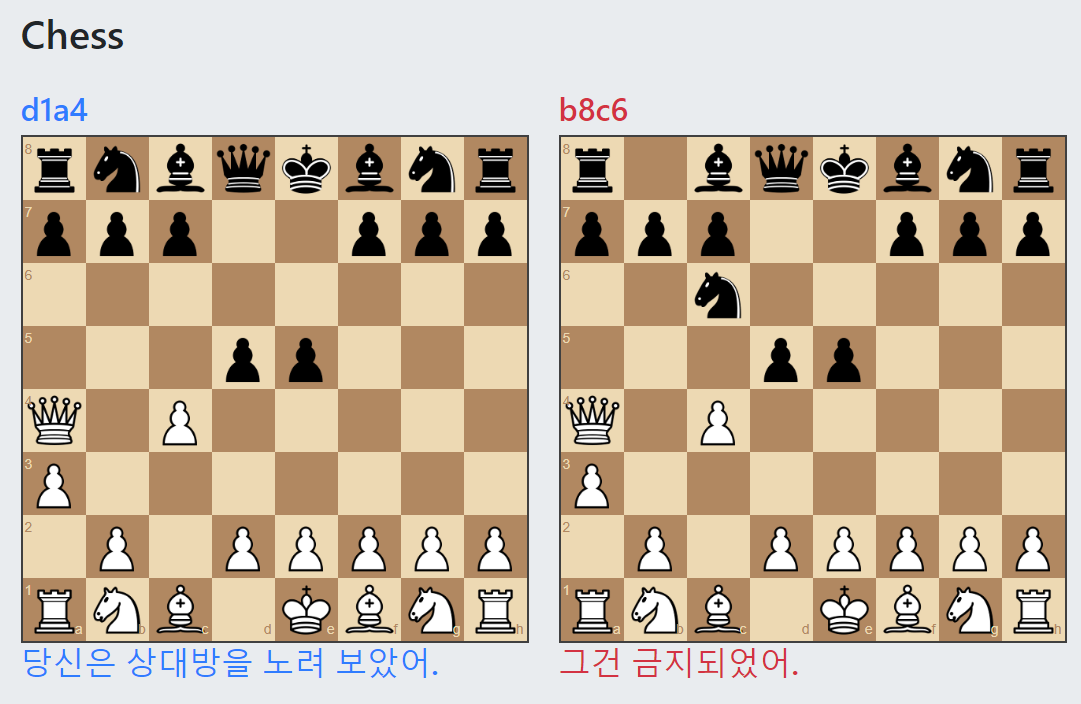


체스를 진행하기 위한 구조이다. game start를 한 후, input으로 player의 move를 받으면 move가 valid한지, invalid한지, engine 돌아가고 있는지를 확인한다. 그 뒤 valid하다면 annotation을 받고, 승리 여부를 판단한 뒤, 승리가 아니라면 컴퓨터가 수를 두고 컴퓨터의 수에 대한 annotation도 받고, 패배가 아니라면 게임을 계속한다.

**7. 구동예시 (Example)**

NUGU play builder를 사용한 예시이다.





플레이어는 스피커 만으로 게임을 진행할 수 있고, 또 시각적인 정보가 필요한 경우 웹사이트에서 체스보드를 실시간으로 보면서 진행할 수도 있다.

**8. 참고자료(References)**

[1] M. Campbell, A. J. Hoane, and F.-h. Hsu, "Deep Blue," *Artificial Intelligence,* vol. 134, no. 1, pp. 57-83, 2002/01/01/ 2002.

[2] O. E. David, N. S. Netanyahu, and L. Wolf, "DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess," Cham, 2016, pp. 88-96: Springer International Publishing.

[3] T. Mikolov, K. Chen, G. s. Corrado, and J. Dean, *Efficient Estimation of Word Representations in Vector Space*. 2013.