JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering
Vol.00, No.0, Jan 0000, pp. 000-000

An Example Usage of the JCSE LaTeX Class

Debra Park
School of Electrical Engineering and Computer Science, ABC National University, Seoul, Korea **debbie@abc.ac.kr**

Dezhen Zhu*
School of Electrical Engineering and Computer Science, ABC National University, Seoul, Korea **jdj@abc.ac.kr**

Abstract
대한민국의 대학수학능력시험 중 생명과학 문제들의 유형을 분류하고, 5개의 최종 노드로 나누어 합성곱신경망(Convolutional Neural Network, CNN) 기반의 해법을 모색한다. 각 문제의 첫 문단을 Tokenizer와 Padding 작업을 거쳐 5,000개의 데이터를 전처리하는 단계를 거친다. 모델 개발 단계에서는 그리드-서치 알고리즘 기법을 이용하였으며, 하이퍼파라미터를 인위적으로 설정하여, 각각의 구간으로 나눈다. 시험 평가 단계에서는 Train-Validation-Test Split (6:2:2)을 통해 결과적으로 729 개의 모델들로부터 각 구간에 대한 정확도를 평가하게 된다. 그 다음에는, Test-Set 평가에서의 정확도가 높은 상위 5개의 모델을 찾아, 모집단의 정확도를 구하기 위해 샘플 모델들을 각각 100 개 씩 만들어 표준정규분포로 정확도를 분석하였을 때, 유의 수준 0.05 내에서 93.71% 의 정확도를 보여준다. 최종적으로, 각 문제들의 텍스트를 인식하여 정해진 5개의 유형에 맞게 자동 분류할 수 있는 AI 모델을 구현하도록 한다.

Category: Embedded Computing

Keywords: CNN, Train-Validation-Test Split, LMS, Data Preprocessing

I. INTRODUCTION

대한민국의 국가 시험 중 대학수학능력시험은 대학을 진학하기 위한 수험생들의 마지막 관문이다. 여러 교육 산업들은 수험생들이 시험 대비를 할 수 있게 도와주면서 새로운 사업 모델(AI) 시장을 모색하기 위해, LMS(Learing Management System)에 AI를 부가 도입하는 등의 행태를 보이고 있다. 현재 수준의 교육용 AI는 수학의 수식 인식을 통한 수학 문제 유형 분류 모델이 개발되었으나, 나머지 과목들 중 텍스트 중심 형태로 이루어진 문제들을 유형 분류하는 모델 개발 연구가 집중적으로 이루어진 사례는 거의 드물다.

이 연구에서는 딥러닝 분야 중 CNN(Convolutional Neural Network)을 통하여 텍스트 전처리를 통해 생명과학 1 유형을 5개로 나누어 분류하는 모델을 개발한다. 데이터는 해당 과목 중 생명과학 5,000개의 문제로부터 각각 첫 번째 문단을 텍스트로 변환 처리한다.

데이터 전처리 단계에서는 토크나이저 작업과 패딩 작업을 거치게 된다. 전처리된 데이터들은 모두 Train Set, Validation Set, 그리고 Test Set으로 나누게 된다.

모델 개발 단계에서는 그리드-서치 알고리즘 기법을 통하여 총 6개의 하이퍼파라미터 독립 변수들(Input_dim, Output_dim, Epochs, Conv1D_filter, Conv1D_kernel, density)을 이용하여 729개의 서로 다른 모델을 생성하게 된다.

평가 시험 단계에서는 각 모델들의 정확도로부터 하이퍼파라미터 구간에 따른 평균-정확도를 구해 분석하게 된다. 그 다음, 정확도가 높은 상위 5개의 모델을 각각 100개씩 생성하여 Z-분포의 신뢰도 90%, 95%, 99%에 따른 정확도를 분석하게 된다.

II. RELATIVE WORK

일반적으로, CNN 모델은 이미지를 인식하여 패턴을 찾는 분류 모델로, 2차원 구조에서 이미지의 공간 정보를 유지하여 Fully Connected Layer에 입력 데이터를 1차원 배열 형태로 한정한다. 하지만 텍스트를 인식하여 분류하는 모델 구현할 경우, 주로 1차원 컨볼루션 망을 이용하여 텍스트의 시퀀스에서 패턴을 찾게 되며 임베딩 작업을 통해 실수 벡터로 변환해 문맥적 특징을 추출하게 된다.

텍스트로부터 word2vec을 통한 전처리 작업(Embedding)을 거치기도 하지만, 해당 작업이 꼭 모델의 성능 향상에 도움을 주진 않는다는 연구 결과[1]가 있어, 복잡한 전처리 과정을 진행하지 않는다.

먼저, 텍스트를 숫자 차원으로 변환하기 위해서는 두

Open Access yy.5626/JCSE.2011.5.2.xxx http://jcse.kiise.org

Received 00 Month 2011, Accepted 00 Month 2011, Revised 00 Month 2011

* Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

가지 과정의 전처리 작업을 거치게 된다.

토크나이저(Tokenizer) 단계에서는 텍스트 문장을 작은 단위인 'Token'으로 나눈 후, 각 토큰에 대해 정수 인덱스 할당하는 과정을 거치게 된다. 단어 대신 인덱스로 교체하여 모델을 학습시키 위한 데이터가 된다.

패딩(Padding) 단계에서는 전 단계에서 만들어진 시퀀스 데이터(Sequence Data)를 일정하게 길이를 맞추는 작업을 하게 된다.

이로써, 첫 레이어에서는 저차원 벡테로 임베드하게 되며, 그 다음 레이어(Hidden Layer)에서는 여러 사이즈의 필터를 통해 벡터에 대한 합성곱 변환을 진행하게 된다.

A. Grid-Search Algorithm

성능(정확도)이 좋은 모델을 개발하기 위해, 개발 모델의 변수들에 인위적으로 정적인 값들을 대입하여 탐색하는 기법[2]을 제안한다. 해당 모델 개발에서 하이퍼 파라미터는 총 6개로, 독립 변수들(Input_dim, Output_dim, Epochs, Conv1D_filter, Conv1D_kernel, density)에 각각 세 구간으로 나누어 여러 조합의 모델을 탐색하게 된다.

B. Train-Validation-Test Split

기존에는 모델의 성능 평가를 위해 데이터셋 분리는 Train-Test Split (8:2) 형태[3]로 진행됐으나, 해당 모델의 개발 과정에서 좀 더 면밀한 관찰을 위해 Train-Validation-Test Split (6:2:2)를 제안한다. 이러한 데이터셋 분리가 도움이 되는 이유는 Validation-Set과 Test-Set 간의 차이가 있다. Validation-Set에서는 훈련시킨 데이터에 대해 모델의 정확도를 검증하게 된다. 반면에 Test-Set에서는 훈련되지 않은 데이터에 대해 모델의 정확도를 검증하기에 중간 검증 과정에서 과적합 영역을 쉽게 찾을 수 있다. 따라서 모델의 실제 성능에 대한 추정치를 분석할 수 있으며, 각 상황에 따른 견고성을 고려해 볼 수 있다.

C. Z-분포(표준정규분포)

실험에서 만들어진 모델들은 예측 정확도에 대한 안정성이 검증된 모델들이 아니기 때문에, 어느 정도의 표준적인 평가 척도가 필요하다. 따라서 샘플 모델들의 예측 정확도에 대해서 Z-분포(표준정규분포)를 제안한다. 표본 개수가 약 30개 이상으로 충분할 때, 중심극한정리에 의거하여 독립적인 확률변수 X의 평균은 정규분포에 가까워진다고 볼 수 있다. 따라서 두 개의 매개변수 중 평균과 표준편차에 의해 그 모양이 결정된다.

$$z = \frac{(x - \mu)}{\sigma/\sqrt{N}} \tag{1}$$

무작위로 생성된 데이터셋들로부터 동일한 하이퍼파라미터를 가진 샘플 모델들을 최대한 많이 생성하여 모집단 모델의 정확도를 구하기 위해, 샘플 모델들의 평균-정확도와 표준편차를 구하는 것이다. 이러한 값들을 구한 후 신뢰구간을 3개로 나누어, 각각 90%, 95%, 99%에 대한 정확도 분포를 구하게 된다.

D. Categorical Crossentropy

다중 클래스 분류 문제에서 널리 사용되는 손실 함수 중 하나이며, 실제 클래스 레이블과 모델의 예측 정확도 확률 분포 간의 차이를 측정한다. 주로 softmax 활성화 함수[4]와 함께 사용된다. 각 클래스들의 확률 분포의 합은 1이 되며, 레이블이 1인 클래스만 로그 확률을 계산하고 나머지는 무시한다.

$$H(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{2}$$

III. MODEL GENERATION

이 영역에서는 모델에 대한 시험 평가를 두 번하게 된다.

A. HyperParameter Test

첫 번째는, 전 단계에서 생성된 729개의 모델을 각 하이퍼파라미터 구간의 평균-정확도를 분석하게 된다. 그리드-서치 알고리즘 기법을 통하여 총 6개의 하이퍼파라미터 독립 변수들(Input_dim, Output_dim, Epochs, Conv1D_filter, Conv1D_kernel, density)을 이용하여 729개의 모델들로부터 Validation-Set 평가에서의 평균 정확도와 표준편차를, Test-Set 평가에서의 평균 정확도와 표준편차를 나타낸 것이다.

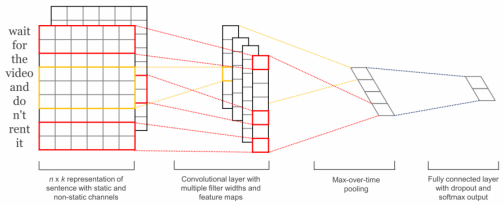

Fig. 1: The CNN Model for text preprocessing

이 영역에서는 텍스트 유형 분류를 위한 모델 개발 단계로, 5,000 개의 전처리된 데이터를 각각 Train-Validation-Test Split으로 나누어 각각 6:2:2로 무작위 Data Set을 생성하게 된다. 각 모델이 만들어질 때마다 모든 Data Set들은 다시 초기화되며, 일련의 작업들을 다시 거치게 된다.

모델 개발 중 Optimizer는 텐서플로우의 케라스 모델로부터 Adam Optimizer로 설정하였다. 학습되는 데이터도 적은 만큼, learning_rate를 평균적인 수치인 0.001로 설정하였다. compile 함수는 'Categorical_Crossentropy'로 설정한다.

그 다음, 그리드-서치 알고리즘 기법을 통하여 총 6개의 하이퍼파라미터 독립 변수들(Input_dim, Output_dim, Epochs, Conv1D_filter, Conv1D_kernel, density)을 이용하여 729개의 서로 다른 모델을 생성하게 된다.

IV. IMPLEMENTATION

Input_dim의 값이 커질수록 모델의 정확도가 크게 향상되고 있다. 이는 더 큰 임베딩 차원을 사용할 때, 모델이

Table I: HyperParameter

Category	Value		
Input_dim	100	150	200
Output_dim	200	300	400
Epochs	5	10	15
Conv1D_filter	64	96	128
Conv1D_kernel	5	10	15
density	64	96	128

Table II: Input_dim

par (N)	100 (243)	150 (243)	200 (243)
avg_v	98.41	99.02	99.43
avg_t	89.88	91.70	93.56
stDev_v	0.79	0.50	0.34
stDev_t	0.84	1.07	0.99

더 좋은 학습 결과를 보이는 것을 알 수 있다. Epochs와 Conv1D_kernel의 값이 커질수록 모델의 정확도가 미세하게 향상되는 것을 알 수 있다. 이는 특성 추출이 잘 개선되었음을 알 수 있다. Output_dim과 density의 값이 모델의 정확도에 유의미한 영향을 주지 못하는 것으로 보인다. Conv1D_kernel의 값이 커지자 검증 세트에서는 정확도가 약간 증가하지만, 테스트 세트에서는 오히려 정확도가 감소하는 현상을 보이고 있다. 과적합(overfitting)의 가능성이 존재한다. 최적의 모델들은 input_dim이 200인 경우, 가장 높은 정확도를 보이고 있다.

A. Standard-Normal-Distribution Test

두 번째는, 모델의 정확도가 높은 상위 5개 모델을 각각 100개씩 무작위로 생성하여 모델 평가를 진행하며, Z-분포의 신뢰 구간(90%, 95%, 99%)에 따라 정확도를 분석하게 된다.

Validation-Set 평가에서의 정확도가 가장 좋은 모델은 [200, 300, 10, 64, 10, 128]이며, Test-Set 평가에서의 정확도가 가장 좋은 모델은 [200, 300, 10, 64, 10, 128]이다. 학습되지 않은 데이터를 넣는 상황 속에서 일반화하기에 좋은 모델은 후자 임을 알 수 있다. 정확도 면에서 Validation-Set 평가에서보다 Test-Set 평가에서의 표준 편차가 큰 이유에 대해서는, 과적합 (overfitting)의 여지가 있으며, 훈련되는 데이터 셋의 크기가 아직도 현저히 작다는 것을 의미한다.

V. CONCLUSION

일반적으로 생명과학이라는 과목에서 5,000개의 데이터를 5개의 유형으로 나눠 학습된 유형 분류 모델에서는 20문제 중 19개의 꼴로 적절한 유형으로 나눌 수 있음을 알 수 있다. CNN 모델의 간단한 유형 분류에서도 텍스트 중심 데이터 학습으로부터 약 93.71%라는 대체로 높은 정확도를 보였다.

ACKNOWLEDGEMENTS

.

Table III: Output_dim

par (N)	200 (243)	300 (243)	400 (243)
avg_v	98.91	98.98	98.97
avg_t	91.68	91.71	91.75
stDev_v	0.76	0.69	0.67
stDev_t	1.79	1.76	1.82

Table IV: Epochs

par (N)	5 (243)	10 (243)	15 (243)
avg_v	98.64	99.07	99.16
avg_t	91.57	91.69	91.88
stDev_v	0.89	0.56	0.49
stDev_t	1.82	1.83	1.71

REFERENCES

[1] 조휘열, 김진화, 윤상웅, 김경민, and 장병탁, "Large-scale text classification methodology with convolutional neural network," in *2015년 동계학술발표회*. 서울대학교 공과대학 컴퓨터공학과; 서울대학교 인문대학 협동과정 인지과학전공; 서울대학교 자연대학 협동과정 뇌과학전공, 2015.

[2] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," *ar5iv*, 2019. [Online]. Available: https://ar5iv.labs.arxiv.org/html/1912.06059

[3] K. Gorman and S. Bedrick, "We need to talk about standard splits," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2786–2791. [Online]. Available: https://aclanthology.org/P19-1267

[4] I. Vasyltsov and W. Chang, "Efficient softmax approximation for deep neural networks with attention mechanism," *CoRR*, vol. abs/2111.10770, 2021. [Online]. Available: https://arxiv.org/abs/2111.10770

Table VIII: 2nd Model Specifications and Predictions

Category	**Sample Model (100 * 5)**				
Input_dim	200	200	200	200	200
Output_dim	300	300	300	300	400
Epochs	5	5	5	10	5
Conv1D_Filter	64	64	64	64	128
Conv1D_Kernel	10	10	10	10	5
density	128	64	64	128	64
Prediction Validation	99.215 (0.326)	99.224 (0.287)	99.426 (0.257)	99.360 (0.246)	98.819 (0.388)
90%	±0.0537	±0.0473	±0.0423	±0.0405	±0.0639
95%	±0.0640	±0.0564	±0.0504	±0.0483	±0.0761
99%	±0.0841	±0.0741	±0.0663	±0.0635	±0.1001
Prediction Test	93.548 (0.866)	93.366 (0.808)	93.546 (0.774)	93.713 (0.935)	93.057 (0.942)
90%	±0.1424	±0.1330	±0.1274	±0.1538	±0.1551
95%	±0.1697	±0.1585	±0.1518	±0.1833	±0.1848
99%	±0.2231	±0.2083	±0.1996	±0.2409	±0.2428

Table V: Conv1D_filter

par (N)	64 (243)	96 (243)	128 (243)
avg_v	98.85	99.09	98.92
avg_t	91.73	91.51	91.90
stDev_v	0.73	0.70	0.67
stDev_t	1.91	1.61	1.83

Table VI: Conv1D_kernel

par (N)	5 (243)	10 (243)	15 (243)
avg_v	98.54	99.22	99.11
avg_t	91.69	91.94	91.51
stDev_v	0.92	0.46	0.43
stDev_t	1.82	1.86	1.65

Table VII: density

par (N)	64 (243)	96 (243)	128 (243)
avg_v	99.05	98.75	99.07
avg_t	91.75	91.60	91.79
stDev_v	0.66	0.79	0.62
stDev_t	1.90	1.59	1.87

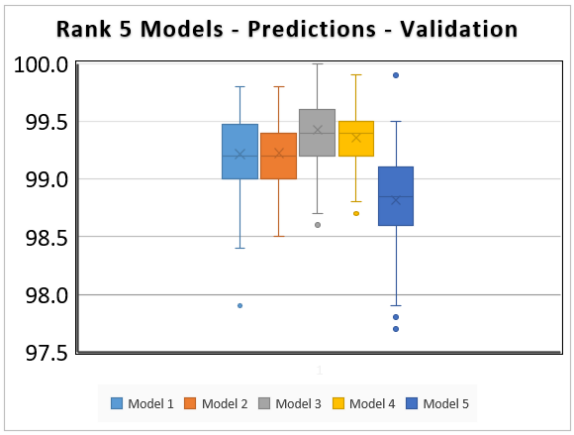

Fig. 2: Rank 5 Models - Validation

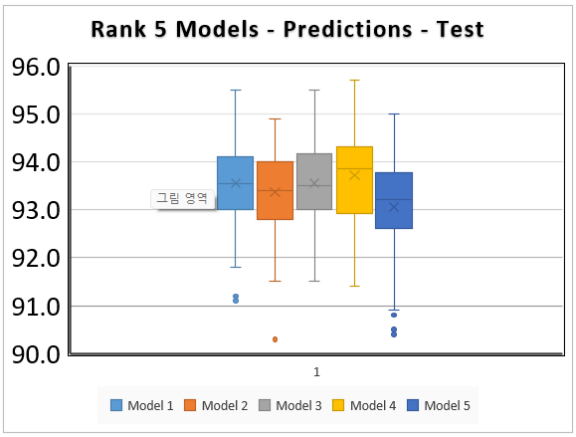

Fig. 3: Rank 5 Models - Test