JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 22, No. 10: 1300~1306, Oct. 2018

딥러닝과 Char2Vec을 이용한 문장 유사도 판별

임근영[1*] · 조영복[1]

The Sentence Similarity Measure Using Deep-Learning and Char2Vec

Geun-Young Lim[1*] · Young-Bok Cho[1]

[1*]Department of Information Security, Daejeon University, Daejon, 34520 Korea

요 약

본 연구는 자연어 처리 문제 중 하나인 문장 유사도 판별 문제를 딥러닝으로 해결하는 데에 있어 Char2Vec기반으로 문장을 전 처리하고 학습시켜 그 성능을 확인하고 대표적인 Word Embedding 모델 Word2Vec를 대체할 수 있는 가능성이 있는지 파악하고자 한다. 임의의 두 문장을 비교할 때 쓰는 딥러닝 구조로 Siamese Ma-STM 네트워크를 사용하였다. Word2Vec와 Char2Vec를 각각 기반으로 한 문장 유사도 판별 모델을 학습시키고 그 결과를 분석하였다. 실험 결과 Char2Vec를 기반으로 학습시킨 모델이 validation accuracy 75.1%을 보였고 Word2Vec를 기반으로 학습시킨 모델은 validation accuracy 71.6%를 보였다. 따라서 고 사양을 요구하는 Word2Vec대신 임베딩 레이어를 활용한 Char2Vec 기반의 전처리 모델을 활용함으로 분석 환경을 최적화 할 수 있다.

ABSTRACT

The purpose of this study is to see possibility of Char2Vec as alternative of Word2Vec that most famous word embedding model in Sentence Similarity Measure Problem by Deep-Learning. In experiment, we used the Siamese Ma-LSTM recurrent neural network architecture for measure similarity two random sentences. Siamese Ma-LSTM model was implemented with tensorflow. We train each model with 200 epoch on gpu environment and it took about 20 hours. Then we compared Word2Vec based model training result with Char2Vec based model training result. as a result, model of based with Char2Vec that initialized random weight record 75.1% validation dataset accuracy and model of based with Word2Vec that pretrained with 3 million words and phrase record 71.6% validation dataset accuracy. so Char2Vec is suitable alternate of Word2Vec to optimize high system memory requirements problem.

키워드 : Word2Vec, Char2Vec, 딥러닝, GRU, NLP

Keywords : Word2Vec, Char2Vec, Deep-learning, GRU, NLP

Received 28 June 2018, **Revised** 9 July 2018, **Accepted** 19 July 2018

*** Corresponding Author** Geun-Young Lim(E-mail:interruping@naver.com, Tel:+82-42-280-2406)
Department of Infromation Security, Daejeon University, Daejon, 34520 Korea

Open Access http://doi.org/10.6109/jkiice.2018.22.10.1300 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

자연어 처리는 최근 심층신경망(Deep Neural Networks, DNN) 기법을 기반으로 지속 발전하고 있다. 자연어 처리에서 먼저 이루어져야 하는 작업은 텍스트를 컴퓨터에 입력할 수 있는 양식으로 변환하는 과정이다. 해당 변환 작업의 예로는 텍스트를 ASCII 코드 또는 UNICODE 등으로 변환하여 처리하는 방법이 있다. 또 다른 방법으로는 단어를 다차원 공간에서 특정한 수치를 갖는 벡터로 사상시키는 Word Embedding이 있다[1]. 그 중에서도 2013년 구글에서 발표된 Word2Vec는 학습속도와 성능을 높혀 인기를 끌고 이다. Word2Vec와 컨볼루션 신경망(Convolution Neural Network, CNN)이 더욱 효과적이라는 것이 알려지면서, 현재는 단어를 벡터(vector)로 표현하는 방법인 Word2Vec과 CNN을 이용한 문장 분류 방법이 제안되고 있으며 CNN을 기반으로 한 다양한 논문이 제시되고 있다. 2017년 국내 연구로 Word2Vec과 CNN기반의 한국어 문서 분류가 제시되었으나 한글에서는 높은 성능을 보이지는 못했다[2]. 본 논문에서는 Word2Vec의 문제점을 해결하기 위해 딥러닝을 이용한 문장 유사도 자연어 처리시 중요한 메모리 문제를 해결하면서 다중어 처리가 가능한 Char2vec 모델 사용의 타당성을 실험을 통해 증명한다[3]. 본 논문에서는 딥러닝 알고리즘으로 해결하는데 Word2Vec와 Char2Vec의 성능을 비교하고 Char2Vec 모델이 Word2Vec 모델을 대체할 수 있는지 가능성을 실험을 통해 증명한다. 본 논문의 구성은 2장 관련연구에서 Word2Vec와 Char2vec에 대해 기술하고 3장에서는 워드임베딩알고리즘의 성능평가를 위한 문장 유사도 판별 실험을 제시한다. 4장에서는 논문의 결과와 향후 연구에 대해 기술한다.

Ⅱ. 관련연구

자연어 처리를 위해 사용되는 워드임베딩 기술로 Word2Vec나 char2vec를 사용하고 있다. Word2Vec은 단어(Word)를 기준으로 하여 주변 단어들을 가지고 그 중심 단어를 예측과 특정 중심 단어를 가지고 그 주변 단어를 예측하는 알고리즘으로 모든 단어를 단일 Char2Vec은 단어보다 더 작은 단위인 문자를 사용하는 기법이다.

2.1. Word2Vec 모델

Word Embedding 알고리즘 중 하나인 Word2Vec는 2013년 구글에서 발표하였으며, 학습속도와 성능을 높혀 인기를 끌고 있는 알고리즘이다. Word2Vec 모델을 통한 단어의 벡터화는 단어의 의미를 수치화하여 나타낸 것임으로 단어 간 유사도를 계산하고 연관 단어 클러스터링을 통하여 이러닝의 수강 기록을 기반 이를 통한 수강 후기를 분류하는 연구가 진행되었었다[4]. 또한 RNN과 같은 딥러닝 알고리즘을 기반의 문서 분류에서 Word2Vec를 활용하는 연구가 진행되었었다[5]. Word2Vec의 LSTM 딥러닝 알고리즘을 Siamese network 형태로 구현하고 이 네트워크 간 맨하탄 거리를 구하여 문장의 유사도를 측정한 연구도 진행 되고 있다[6].

2.2. Word2Vec를 활용한 클러스터링

Word2Vec 알고리즘으로 단어를 벡터로 변환하게 될 시, 단어의 의미가 수치화 되어있어 벡터 연산을 통해 수치적인 접근이 가능하다. Word2Vec 모델을 학습시킬 경우, 학습한 모든 단어에 대한 feature 벡터를 학습하게 된다. 임베딩 할 단어의 개수를 V, 은닉 층의 차원 수 가 N이라 할 때 가중치 행렬은 $V \times N$ 크기를 가지게 된다. 즉, 임베딩 할 단어의 개수에 비례해서 가중치 행렬의 크기가 증가한다. 구글에서 공개한 Word2Vec모델은 3백만 개의 단어로 학습되었고 300차원의 가중치를 갖는 모델이다. 이 모델의 크기를 계산하면 식1과 같다. 식1에서 N을 단어의 개수, V를 벡터의 차원 수, B를 가중치 바이트 크기를 의미한다.

$$\frac{N \times V \times B}{1024^2} \fallingdotseq Word2Vec\ size \\ = \frac{3000000 \times 300 \times 4}{1024^2} \fallingdotseq 3.4\ GB \tag{1}$$

Char2Vec는 Word2Vec와 달리 단어가 아닌 문자를 벡터로 변환하는 모델로 Word2Vec와 비교했을 때 상대적으로 적은 용량을 가지게 된다. Word2Vec의 경우 학습시키는 단어의 개수에 따라 모델의 크기가 결정되지만 Char2Vec 모델은 문자의 개수크기 만큼의 메모리를 요구하게 된다.

2.3. Siamese Ma-LSTM 문장 유사도 판별

가변길이의 시퀀스의 쌍으로 구성된 데이터를 활용하기 위한 LSTM 딥러닝 네트워크 모델로 문장들 간의 의미론적 유사성을 평가하기 위해 해당 Siamese Ma-LSTM 네트워크가 적용된다. 네트워크의 성능을 실험해보았을 때 사람의 손으로 만들어진 Feature로 구성된 복잡한 신경망 시스템보다 우수한 성능을 보이나 LSTM은 고정 된 크기의 벡터를 사용하여 문장에서 표현 된 기본 의미를 인코딩한다. 후속 작업을 단순한 Manhattan metric에 의존하도록 제한함으로써 모델에서 배운 문장 표현이 복잡한 의미론적 관계를 반영하는 고도로 구조화 된 공간에 형성하도록 제시하고 있다. 이 연구에서 또한 Word2Vec의 임베딩 레이어를 구현하여 모델이 많은 메모리공간을 요구하고 있다.

2.4. Char2Vec모델

Char2Vec 모델은 Word2Vec 다르게 '문자' 수준에서 작동한다. 따라서 Char2Vec 모델은 맞춤법 오류에 훨씬 더 관대하며, 트위터와 사용자 리뷰 등의 분석에 좋은 성능을 보였다[7]. 또한 국내에서 열린 naver 주최의 AI 해커톤 대회에서 질문 유사도 판정 1위를 기록한 모델 또한 Char2Vec모델을 사용함으로 메모리 관점에서 자연어 처리시 메모리 효율성을 가져온 워드임베딩 모델로 이슈가 되고 있다 [8].

Ⅲ. Word2vec와 Char2vec를 이용한 문장 유사도 판별 실험

본 논문에서는 실험을 위한 절차는 Word2Vec와 Char2Vec를 이용한 전처리 후 전처리 데이터에 맞는 하이퍼 파라미터로 각 Ma-GRU 네트워크에 입력한다. 또한 Word2Vec와 Char2Vec 기반 모델 각각을 200 epoch을 학습을 수행함으로 네트워크 모델의 유사도 판별실험을 수행하였다.

3.1. 데이터 셋

머신러닝/딥러닝 데이터를 공유하고 공유 된 데이터를 두고 경쟁하는 Kaggle에서는 문장 유사도 판별을 위한 Quora의 데이터가 공유되었다. 본 연구에서는 이 Quora 문장 데이터를 사용하여 문장 유사도 판별 딥러닝 모델을 학습시켰다. 표1은 학습에 사용된 데이터의 특징을 설명한 것이다.

Table. 1 Quora Question Pairs Training Dataset Specification

Data Feature	Value
Total Number of rows	404,290
Training Data Number of rows	323,432
Validation Data Number of rows	80,858
Data fields	id
	qid1
	qid2
	question1
	question2
	is_duplicated

표1에서와 같이 404,290쌍의 문장 데이터로 구성되어 있으며, 문장마다 id가 부여되어 있다. is_duplicated가 문장의 유사도 Label이며 1일 경우 유사 문장, 0일 경우 다른 문장이다. 딥러닝 모델의 학습 성능 확인을 위해 데이터셋을 학습데이터, 검증데이터로 8:2 비율로 무작위로 섞은 이 후 나누었다. 따라서 학습데이터 문장 쌍은 323,432개 이며 검증데이터 문장 쌍은 80,858개이다.

3.2. 실험 환경

실험을 위한 프로그래밍 언어 환경은 Python 3.6.2를 사용해 구현하였고, Siamse Ma-LSTM 모델을 구현을 Tensorflow-gpu 1.9 버전을 사용하였다. 또한 문장 데이터 처리를 위하여 numpy 1.15.0과 pandas 0.23.4 버전을 사용하였고, Word2Vec 모델을 사용을 위하여 gensim 3.5.0을 사용하였다. 문장을 단어로 나누는 작업인 tokenize 작업에는 nltk 3.3을 사용하였다. 실험에 사용된 컴퓨터의 사양은 표 2와 같다.

Table. 2 Computer Hardware Specifications for Experiment

H/W	Description
CPU	Intel(R) Core(TM) i3-8100 CPU @ 3.60GHz (4 CPUs), ~3.6GHz
GPU	NVIIDIA GeForce GTX 1060 6GB
Memroy	8192MB

3.3. 유사도 판별을 위한 문장 전리기 과정

Word2Vec 모델은 Google이 공개한 미리 학습된 모델을 기반으로 총 3백만 개의 단어가 학습, 단어 당 300차원의 벡터 가중치를 갖는다. 그림1(A)은 Batch 단위로 문장을 단어로 나눈 다음 벡터화 시켜 딥러닝 모델에 입력하는 방식으로 구현하였다.

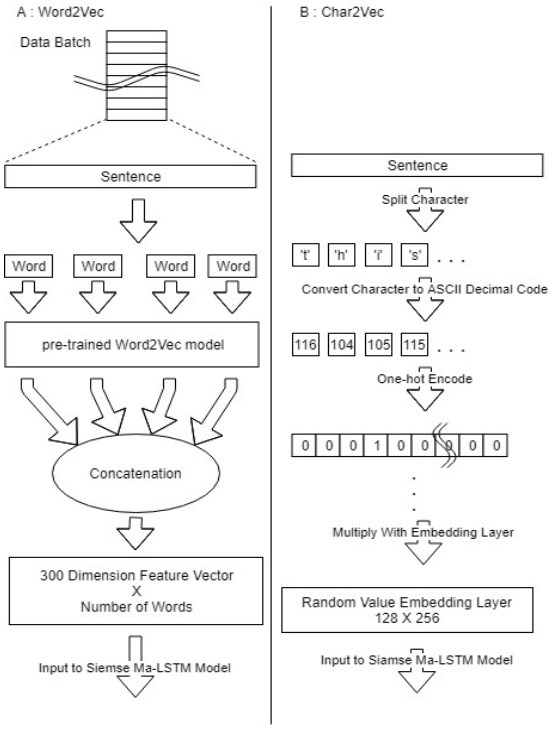

Fig. 1 Text Data preprocess comparison Word2Vec and Char2Vec

Char2Vec는 각 영어 알파벳 문자에 대한 256차원의 벡터를 Embedding Layer로 구성한다. 그림1(B)는 Char2Vec 모델이 문장을 처리하는 과정을 도식화한 것이다. 입력은 ASCII 코드의 십진수 값을 입력으로 하였다. ASCII 코드에 없는 문자는 일괄적으로 ASCII 코드 십진수 값의 최대값 보다 1큰 128로 처리하고 ASCII 코드 십진수 값은 One-hot 인코딩 되고 Embedding Layer랑 행렬 곱을 수행한 뒤, 입력 문자 코드에 해당하는 256차원의 Embedding 된 벡터가 최종적으로 딥러닝 모델에 입력되어진다. Char2Vec의 Embedding Layer의 가중치 값들은 -1.0과 1.0 사이의 무작위한 실수 값으로 초기화 하였다.

3.4. Siamese Ma-LSTM 모델 구현

Word2Vec 또는 Char2Vec로 전 처리가 끝난 데이터는 Siamese 아키텍처로 구성된 Ma-LSTM으로 입력된다. 문장 유사도 측정에 LSTM을 사용을 위해 본 연구에서는 LSTM의 장점을 유지하면서 계산복잡성을 낮춘 GRU를 사용하였다[9]. RNN의 timestep을 결정하기 위해, 데이터 셋의 문장의 문자 개수와 단어 개수의 분포를 알아보았다. Word2Vec의 timestep은 20, Char2Vec와 마찬가지로 짧은 데이터의 남는 부분은 zero padding으로 처리하였다. 그림 2은 질문1과 질문2의 유사도를 계산하는 딥러닝 네트워크의 구조로 Char2Vec와 Word2Vec로 문장을 전 처리시 timestep과 feature 벡터의 차원 수만 차이가 나고 다른 부분은 동일하다. 질문1과 질문2의 입력을 같은 GRU RNN으로 계산한다. Jonas Mueller의 연구에서와 같이 두 질문의 결과 벡터의 Manhattan distance를 계산한다[6]. 유사도 함수는 e^{-x} 곡선을 따르게 되고 거리가 0에 가까울 수 록 결과 출력 y는 1, 거리가 멀수록 출력 y는 0에 수렴하게 된다.

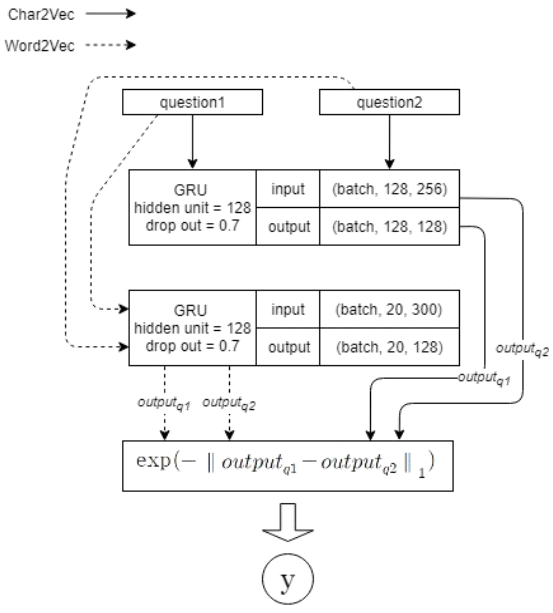

Fig. 2 Compare Word2Vec with Char2Vec of Siamse Ma-GRU Architecture

3.5. 딥러닝 모델 학습

Word2Vec와 Char2Vec 두 실험 케이스 모두 학습 Batch 사이즈는 512로 설정하고 200 epoch을 학습하였다. 검증 데이터 셋의 계산은 100 step마다 수행하고 검증 데이터 셋의 정확도 측정 metric은 0.5를 threshold로 설정 후 true/false를 판정하였다. 손실 함수의 metric은 (2)와같이 log loss를 사용하였다[10].

$$-(y\log(p)+(1-y)\log(1-p)) \quad (2)$$

제안 모델의 학습 최적화 알고리즘은 Adam[11]을 사용하였으며 학습률은 0.0001값을 사용하였다. 3.3에서 언급하였듯이 Word2Vec는 모든 단어에 대한 vector들을 Embedding Layer로 구성하는 것이 실험 컴퓨터 메모리 한계로 가능하지 않았고, 그래서 Batch 단위로 vector 변환을 하였다. 따라서 Word2Vec 실험 케이스에서 200 epoch 학습에 약 20시간이 소요되었고 이는 Char2Vec 케이스의 12시간보다 학습시간이 8시간 더 오래 걸린 것이다.

Ⅳ. 결 과

우리는 Word2Vec와 Char2Vec의 실험 케이스를 구분하여 Char2Vec의 성능을 살펴보았다. Word2Vec의 실험의 경우 미리 학습된 모델을 사용한데 반해 Char2Vec는 Feature Vector를 무작위한 값으로 초기화하고 학습을 진행하였다.

4.1. 학습결과

Char2Vec와 Word2Vec의 학습 결과는 각각 그림3와 그림4의 그래프에서 볼 수 있다. 200 epoch의 학습을 수행 한 결과, Char2Vec 기반의 질문 문장 유사도 측정 모델은 검증 데이터 셋 정확도 75.7%를 기록하였다. Word2Vec 기반의 질문 문장 유사도 측정 모델은 검증 데이터 셋 정확도 71.6%를 기록하였다.

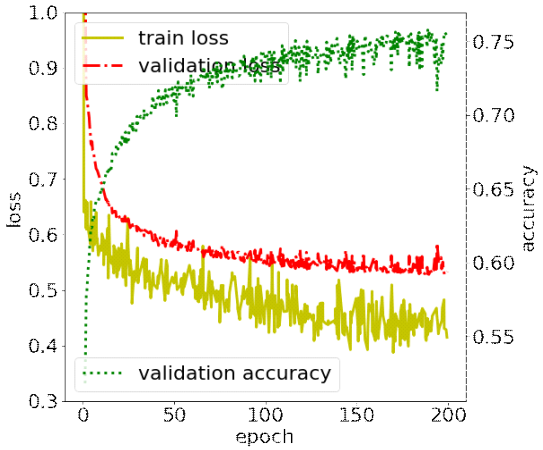

Fig. 3 Char2Vec model based training graph

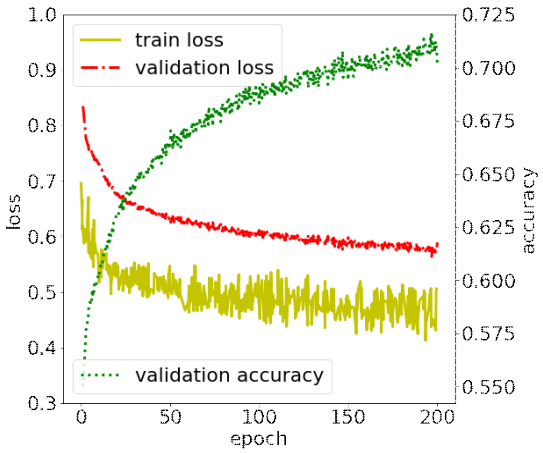

Fig. 4 Word2Vec model based training graph

Char2Vec 모델의 경우 Embedding Layer의 가중치 값을 -1.0과 1.0 사이의 무작위 한 값들로 초기화 한 것임에도 Word2Vec 기반 모델과 비슷한 성능을 보여준 것은 눈에 띄는 점이였다. Char2Vec와 Word2Vec 기반 모델의 학습 그래프를 관찰하였다.

Table. 3 Standard deviation comparison (unit :std-dev)

	train loss	validation loss	accuracy loss
Char2Vec	**0.1902**	**0.0604**	**0.0344**
Word2Vec	0.0399	0.0452	0.0318

표 3에서 오차율과 정확도의 표준편차를 계산을 통해 Char2Vec 모델의 학습 진행에 있어서 Word2Vec보다 약 0.15 큰 것을 확인할 수 있었고 Char2Vec가 학습이 진행될수록 변화의 폭이 커지면서 불안정한 모습을 보임을 알 수 있었다.

Ⅴ. 결 론

본 연구에서는 자연어 처리를 위해 대중적으로 사용되고 있는 Word2Vec가 갖는 문제점을 해결하기 위해 동일 성능으로 사용가능한 Char2Vec의 사용 타당성을 실험을 통해 증명하였다. 자연어 처리의 문장 유사도 측정을 위해 구글의 Word2Vec가 많이 사용되고 있으나 Word2Vec 처리를 위해서는 높은 메모리 사용량이 요구된다. 또한 한글과 영문 등이 언어가 혼용된 다중어 처리가 어렵고 한글에서는 영문보다 낮은 성능을 보이고 있다. 따라서 이 논문에서는 이런 문제를 해결하기 위해 워드임베딩 방법 중 메모리 효율성을 높이고 다중어 처리가 가능한 Char2Vec으로 대체가 가능한지 실험을 통해 증명하였다. 실험결과, Char2Vec 기반의 질문 문장 유사도 판정 딥러닝 모델을 기존 범용적으로 사용되는 Word2Vec기반 모델과 비교했을 때 학습 과정에 있어서 상대적으로 다소 불안정하였으나 성능측정에서 봤을 때 Char2Vec를 기반으로 학습시킨 모델이 validation accuracy 75.1%을 보였고 Word2Vec를 기반으로 학습시킨 모델은 valid accuracy 71.6%를 보였다. 따라서 자연어 처리 문제를 처리함에 있어 딥러닝 알고리즘과 접목하는 경우 Word2Vec을 대신해 Char2Vec모델로 대체할 수 있는 가능성이 있음을 보였다. 실험 결과 Word2Vec 실험의 경우 미리 학습된 모델을 사용하는 반면 Char2Vec는 Feature Vector를 무작위한 값으로 초기화하고 학습을 진행하였다. 향후 문자들 사이에 벡터를 Word2Vec의 단어 벡터와 같이 특정한 의미가 있는 수치로 학습시킬 수 있는 방법과 한글 및 다중어 처리를 위한 연구가 지속된다면 지금보다 더 우수한 성능을 기대할 수 있을 것이다.

ACKNOWLEDGEMENT

This research was supported by the CHUNGBUK TECHNOPARK, Korea, under the (Development of Prediction and Diagnosis System for Pediatric Adolescents Using Iris based Image Mining) support program (No.20180186)

REFERENCES

[1] S. J. Park, S. M. Choi, H. J. Lee, J. B. Kim, “Spatial analysis using R based Deep Learning,” *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology*, vol. 6, no. 4, pp. 1-8, April 2016.

[2] J. M. Kim and J. H. Lee, “Text Document Classification Based on Recurrent Neural Network Using Word2vec,” *Journal of korean Institute of Intelligent System*, vol. 27, no.6, pp. 560-565, Jun. 2017.

[3] P. Baudiš, S. Stanko and J. Šedivý, “Joint Learning of Sentence Embeddings for Relevance and Entailment,” in *The Workshop on Representation Learning for NLP*, Berlin, Germany, pp. 18-26, 2016.

[4] J. Y. Kim and E. H. Park, "e-Learning Course Reviews Analysis based on Big Data Analytics," *Journal of the Korea Institute of Information and Communication Engineering*, Vol. 21, No. 2, pp. 423-428, Feb. 2017.

[5] J. M. Kim and J. H. Lee, "Text Document Classification Based on Recurrent Neural Network Using Word2vec," *Journal of Korean Institute of Intelligent Systems*, Vol. 27, No. 6, pp. 560-565, Dec. 2017.

[6] M. Jonas, and A. Thyagarajan. "Siamese Recurrent Architectures for Learning Sentence Similarity," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Arizona, pp. 2786-2792, 2016.

[7] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, “Character-Aware Neural Language Models,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* ,Arizona, pp. 2741-2749 , 2016.

[8] Naver ai hackerton 2018 Team sadang solution [Internet]. Available:https://github.com/moonbings/naver-ai-hackathon-2018.

[9] R. Dey and F. M. Salem. “Gate-variants of gated recurrent unit (GRU) neural networks,” in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems*

(MWSCAS), Boston, pp. 1597-1600 , 2017.
[10] wiki fast .ai Logloss [Internet]. Available: http://wiki.fast.ai/index.php/Log_Loss
[11] D. P. Kingma, J. Ba, “Adam: A Method for Stochastic Optimization,” in *The 3rd International Conference for Learning Representations*, pp. 1-15, San Diego, 2015.

임근영 (Geun-Young Lim)

2013-2018 :대전대학교 정보보안학과 학부4학년

※관심분야 : Deep-learning, NLP, Computer Vision

조영복(Young-Bok Cho)

2005: 충북대학교 전자계산학과 공학석사
2012: 충북대학교 전자계산학과 공학박사
2016: 충북대학교 의학과 박사과정수료
2012-2018: 충북대학교 소프트웨어학과 초빙교수
현 재 : 대전대학교 정보보안학과 조교수

※관심분야: 의료영상처리, 정보보안, 의료정보보호, 모바일보안