주제랑 데이터셋, 구현 방법 (모델과 튜닝 부분) 정해놓고 21일 발표회 보고 나서 작성하자

**10 / 26 발표**

: 1 ~ 3 슬라이드 ( 3분 짜리임 )

1. 주제 관련된 논문보셈
2. 논문의 요약, 내가 고른 핵심 아이디어 위주
3. 개발 계획과 향상방향

**프로젝트 보고서 아이디어**

* 주제 선정 요건
* 새로운 점, 아이디어
* 재사용 할만한 것
* 오류와 수정이 요구되는 부분
* 동일 주제의 다른 논문과 비교 (가능하면)
* 논문의 도전과제, 미래의 변경 지향점 시도
* 실효적인 계획을 짜자 ( 기간은 한달임 )
* 데이터 셋 구해오기, 가능한 평가 방법
* 원본과의 변경된 나만의 버전을 비교하라 ( 향상이 있음을 보여줘야함 )

**프로젝트 종류**

* 새로운 작업에 대한 기존의 모델을 쓰던, 뭐하건 상관없음
* 나는 있는 모델에 대해서 튜닝을 하려고 함, 알아보고서 나올거고 이건…

Deep CNN ?

Hello My name is Dong Hwan Kim,

What I’m going to do is about a NLP model for sarcasm detection from the context.

**Research Paper**

* Pelser, D., & Murrell, H. (2019). Deep and Dense Sarcasm Detection. *arXiv preprint arXiv:1911.07474*.

: [1911.07474.pdf (arxiv.org)](https://arxiv.org/pdf/1911.07474.pdf)

* Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., ... & Tang, X. (2017). Residual attention network for image classification. In Proceedings of the IEEE conference on computer vision and

https://arxiv.org/pdf/1704.06904v1.pdf

* Reference Code

: [Dev-data-fun/D-D-Sarcasm: MSc computational sarcasm project, written in Pytorch and FastAI. (github.com)](https://github.com/Dev-data-fun/D-D-Sarcasm)

**Data Set** : [Sarcasm on Reddit | Kaggle](https://www.kaggle.com/danofer/sarcasm?select=train-balanced-sarcasm.csv)

**Summary**

: The sarcasm models usually take the context of a certain dataset ( corpus ), drag the feature out of those contexts using various types of NLP models. The previous sarcasm detection models were focusing on the context and meta-data, without the understanding of main resources in the sarcasm itself. This paper is focused on the features of certain utterances that require background knowledge and commonsense reasoning. In the implementation, they build a block of deep densely connected CNN with three other variations, about isolating utterances and extract richer features therein and trained, tested on News Headlines.

**Plan and Method**

: The idea of this model from the paper is good to be reused in other datasets, among all the other models that this paper were using, they have common sense which, mainly with the feature concentration on the utterances. On the contrary, the feature is extracted well in terms of reasoning matter with the idea of dense blocks to dig out the features is pretty much acceptable, so my plan is to use this models’ feature extraction adding more layers of attention, bottleneck.

using Deep ResNet, about 50 layers. Originally it will take tremendous time to learn. But, by the use of a bottleneck it would need significantly less computation ( because of parameters ) when it comes deeper into ResNet, it will enable our model to be trained faster and cost-effectively.

But using bottleneck has channel increment and decrement, which will cause data loss and decrease of accuracy so the adding Attention Module between Residual Units will be the compensator of our model.

This model will be trained and tested on the SNS (since most of the sarcasm is used in dialogs ) and compared with the other original from the paper. By the idea of transfer learning from reasoning, the confusion of classic sarcasm detection will be a good way to start.

**11/30 프로젝트 보고서**

2, 2, 5, 5 = Original + Dropout 0.2 + Leaky 0.02

train loss 0.14/ valid loss 0.41

acc, macro, micro 0.85/ 0.85/ 0.85

2, 2, 5, 5 → AdaptiveMaxPool1d

train loss 0.17 / valid loss 0.35

acc, macro, micro 0.85/ 0.85/ 0.85

3, 3, 5, 5

train loss 0.23 / valid loss 0.36

acc, macro, micro 0.85 / 0.84 / 0.85

3, 3, 5, 5 + Dense { Down Sample }

train loss 0.24 / valid loss 0.38

acc, macro, micro 0.83

3, 3, 5, 5 + Attention

train loss 0.2 / valid loss 0.36

acc, macro, micro 0.85/ 0.84/ 0.85

→ Dense + Attention 조합으로 Overfitting이 발생함 → Dropout 필수였고 현재까진 더 Overfitt 없음

→ Downsample 의 사용으로 특징 추출의 효과가 부가 되는 것으로 추정.

→ bottleneck 경우 너무 다운 샘플 되는 것 같음 / 사실 transition 이 bottleneck이어서 이중 적용되는 경향

3, 3, 6, 6

train loss 0.15/ valid loss 0.34

acc, macro, micro 0.86/ 0.85/ 0.86

4, 4, 7, 7

→ Overfit 의 조짐 없이 상당한 성능의 항샹, 레이어 추가 증설 // 오히려 Underfit 되버림

train loss 0.25 / valid loss 0.35

acc, macro, micro 0.84/ 0.84/ 0.84

4, 4, 7, 7 + Dropout 0.3 + Leaky 0.015

→ Overfit 의 정황은 존재함, 하지만 어디까지나 관측 횟수가 15회이기 때문일 수 있음

train loss 0.09 / valid loss 0.39

acc, macro, micro 0.85/ 0.85/ 0.85

4, 4, 7, 7 + Dropout 0.25 + Leaky 0.01

→ 0.22 / 0.35

.85 / .85

Local Minima 가 아니라 계속 수렴하는 부분이 Global Minima 임을 파악함, 여러번의 실험 끝에

Dropout, leaky 지수 또는 momentum, 학습횟수 까지 하나의 Local Minima 가 존재하지만 이후에 Global Minima에서 큰폭으로 발산하여 벗어나는 것을 확인함, Lr 조정이 필요 또는 Momnetum 조정이 필요

3, 3, 6, 6 / Dropout 0.3

→ 0.18 / 0.48

이거 보단 Drop 드가야할듯? Leaky 0.01 의 성능이 꽤나 괄목할만 하다.

중간 세팅 점검 –

Dense = True / Merge

Self Attention = True

3, 3, 6, 6 / Dropout 0.35, leaky = 0.01

ReLU to LeakyReLU

AdaptiveAvgPool1d → AdativeMaxPool1d

Epoch 30 언저리에서 새로운 Minima point가 보이는듯 하다

35회차에서 새로운 Min point 발견,

성능은 삐까뜨기 때문에 패치 함.

큰틀은 재정의 되었기 때문에 더 이상은 무의미할것으로 예상

미세 조정을 위해 Lr 과 모멘텀 범위를 건드리기로.

3, 3, 6, 6 // Mom 0.8 ~ 0.7

→ 0.17 / 0.34 / 0.86 / 0.86

오 이렇게 보니까 차이가 있네?

3, 3, 6, 6 // Mom 0.8 ~ 0.7

→ 0.16/ 0.38 / 0.85/ 0.85

모멘텀 0.85 ~ 0.75에서 조작중.

Dropout 0.4

3, 3, 6, 6 // Moms 0.85 ~ 0.75

→ 0.1 // 0.33 / 85 / 85

Dropout 0.35

3, 3, 6, 6 // Moms 0.85 ~ 0.75

→ 0.14 // 0.35 // 84

3, 3, 6, 6

Self Attention = True

Leaky = 0.01

Transition = Densed

Dropout = 0.35

Weight init with 0.5

Momentum 0.85, 0.7

→ 0.18 / 0.33/ 0.86 / 0.86

→ 0.11 / 0.34 / 0.86 / 0.86

# 2, 2, 3, 3, 4

→ 적게 하는게 유리할까? Loss 의 이유는 Downsampling 횟수? = 결국 Downsample은 loss 손실이 옴

→ 많이 하려면 역시 Layer 를 증설하는 것이 훨씬 나아 보임.

train loss 0.04 / valid loss 0.86

acc, macro, micro 0.82 / 0.82 / 0.82