NLP Project:
소설 작가 분류 AI 경진대회

김채윤, 김민아, 백승재, 이유원, 이재리

목차

Project Review

001 >> **프로젝트 주제 및 설명**

[소설 작가 분류 AI 프로젝트]
:: DACON
:: 문체 분석 알고리즘 개발

:: 소설 속 문장뭉치 분석을 통한 저자 예측

002 >> **프로젝트 배경**

•a. 작가의 글을 분석하여 특징 도출
•b. 취향 추천 시스템 활용 / 대필, 유사작 탐지

003 >> **사용 데이터**

• train.csv : 소설 text, autor(0~4)
:: 5명의 저자가 쓴 text data

Project Review

>> Data Augmentation란?

[사전적 의미]
: 데이터의 양과 다양성을 확보하는 방법

[통상적 의미]
: 원본 데이터의 Label을 보존하면서 원본데이터로부터 새로운 데이터를 생성하는 방법

" 데이터 전처리 >> 정제된 전처리 >> 탐색적 자료 분석 >> 모델 >> 사용 "

>> Data Augmentation를 하는 이유

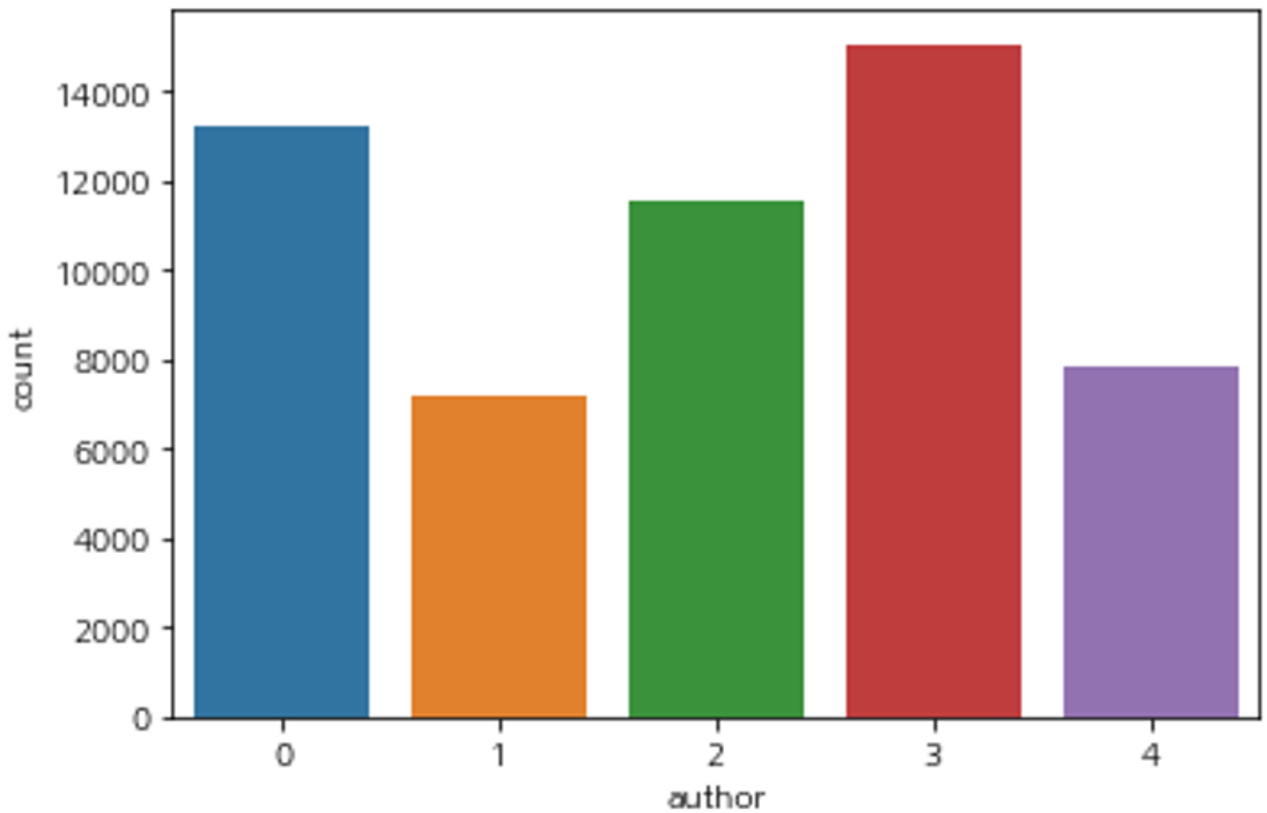
- 문제 해결을 위해서는 데이터 이해가 선행되어야 함
- 다양한 NLP 분야에 활용되어 성능을 향상시키는데 기여함.

NLP - EDA

작가별 데이터 건수	Text 길이	품사 구분	단어 구분	불용어 처리	긍정/부정
이재리	이재리	이유원	김채윤	백승재	김민아
	작가별 문장 길이 소설 평균 길이	작가별 형용사/ 부사 / 동사/ 명사 등 품사 사용 빈도 데 이터 구분	단어의 개수 단어 평균 길이 전체 데이터 최빈 단어 작가별 최빈 단어	Stopwords 제거 Lemmatization	

NLP - EDA

>> 작가별 데이터 개수 확인

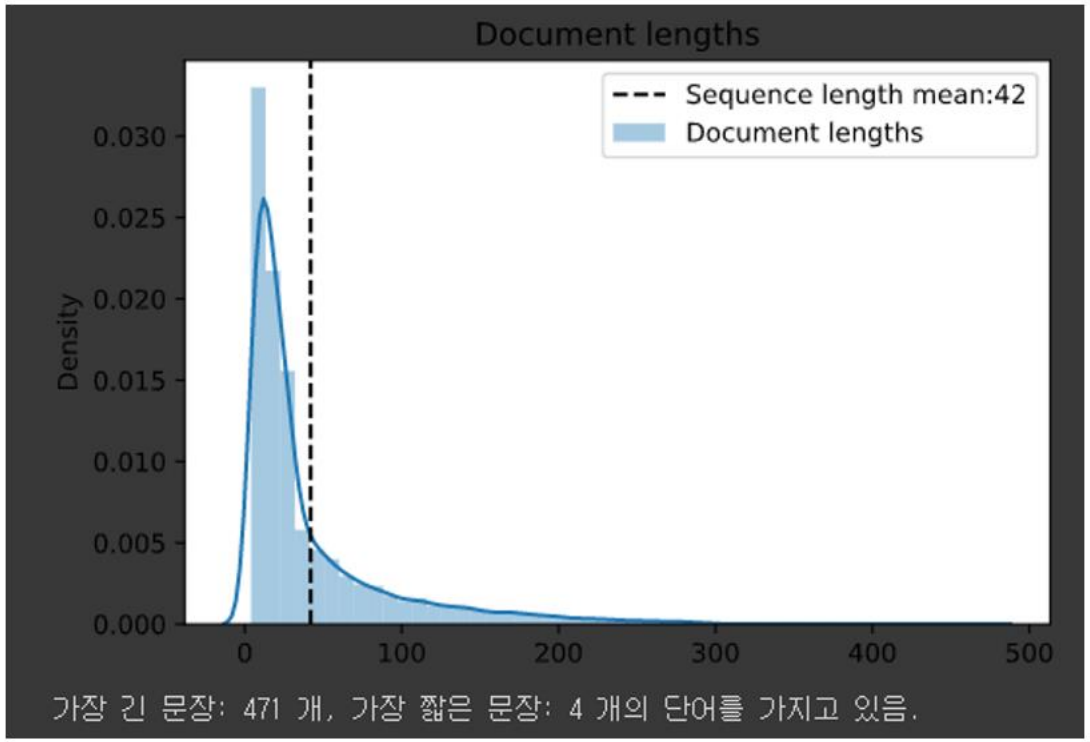


author	index	text
0	13235	13235
1	7222	7222
2	11554	11554
3	15063	15063
4	7805	7805

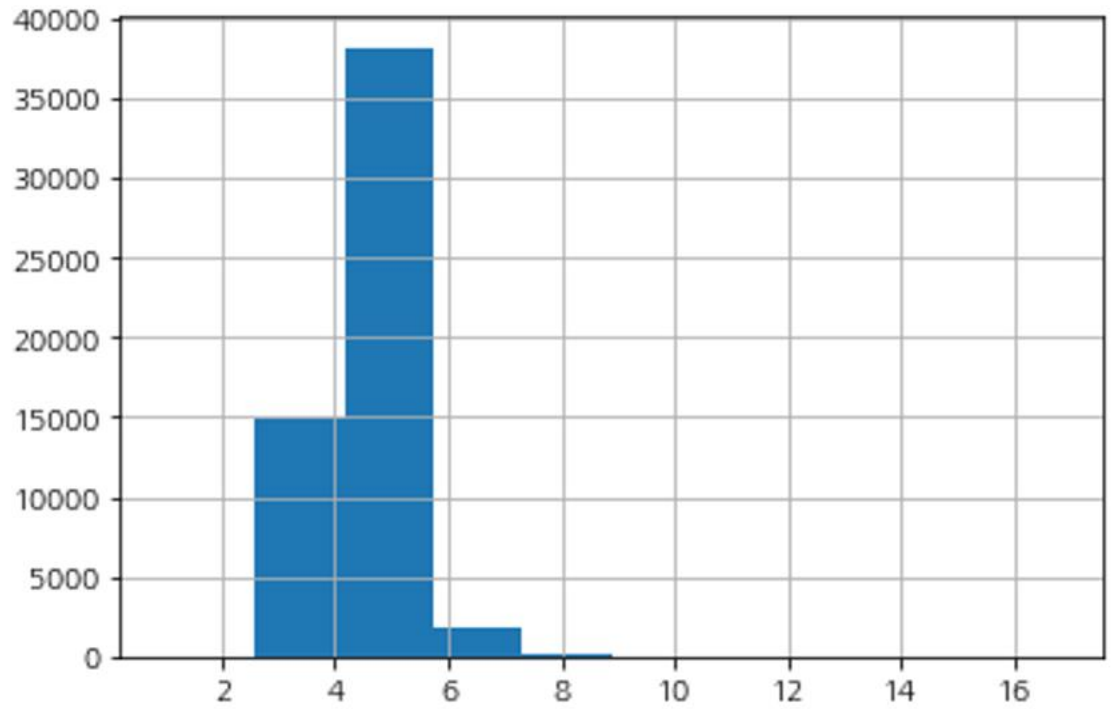
[작가별 소설 개수]

NLP - EDA

>> 전체 데이터 문자/단어의 길이 비교



[문장의 길이]

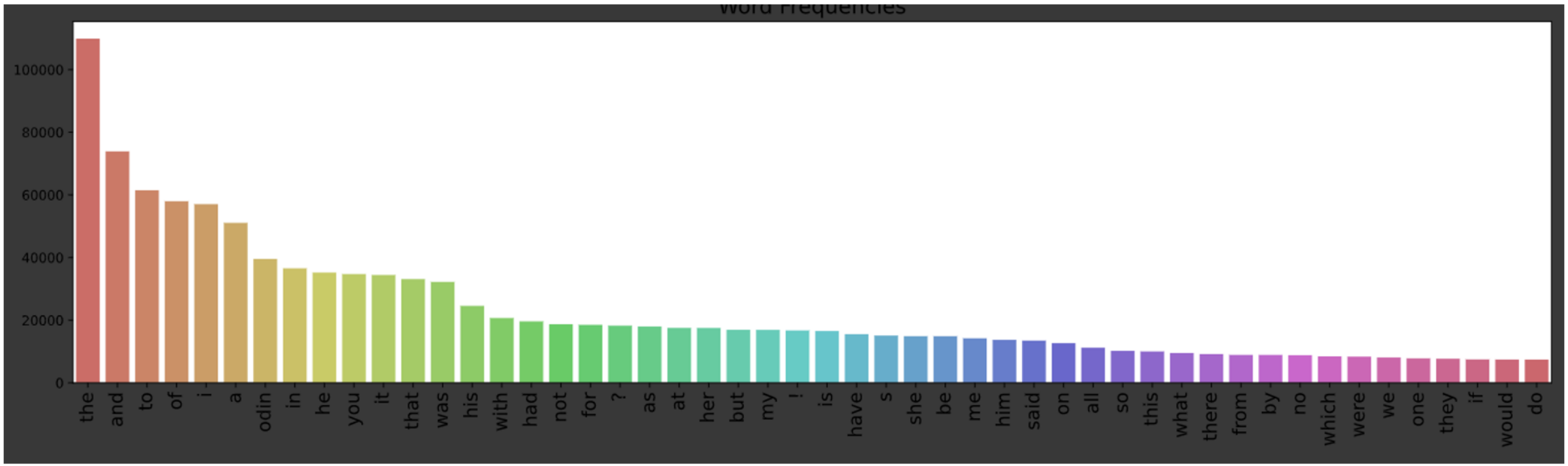


[단어의 길이]

NLP - EDA

>> 단어의 빈도

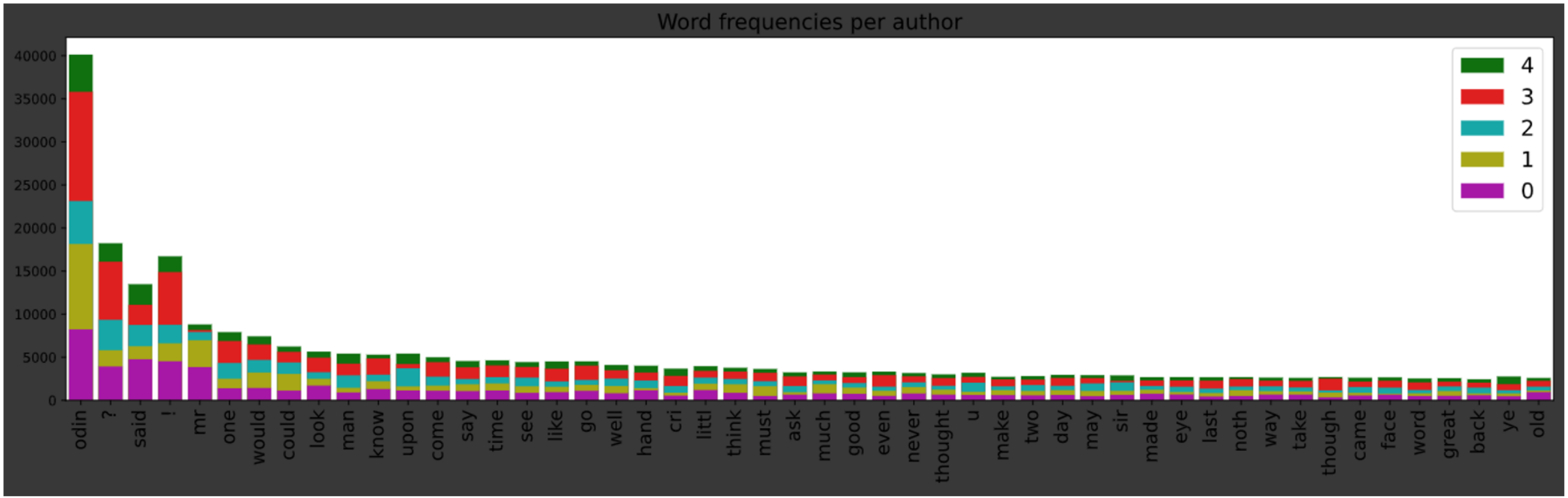
불용어 제거 후 전체 데이터 중 최빈 단어 **Top 50**



NLP - EDA

>> 작가별 단어 빈도

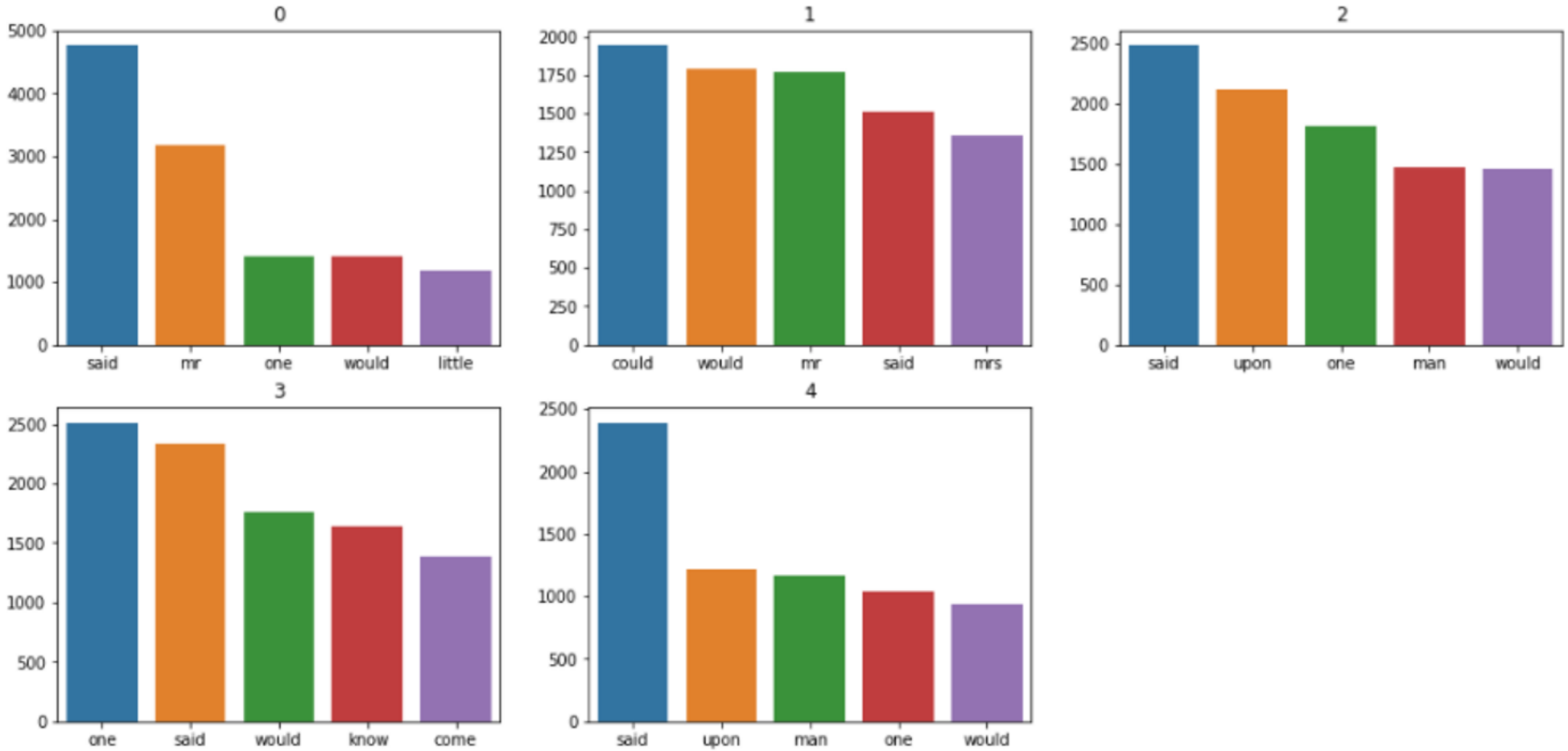
전체 데이터 중 작가들의 단어 사용 빈도



NLP - EDA

>> **Q. 작가별로 최빈 단어가 다를 것이다.**

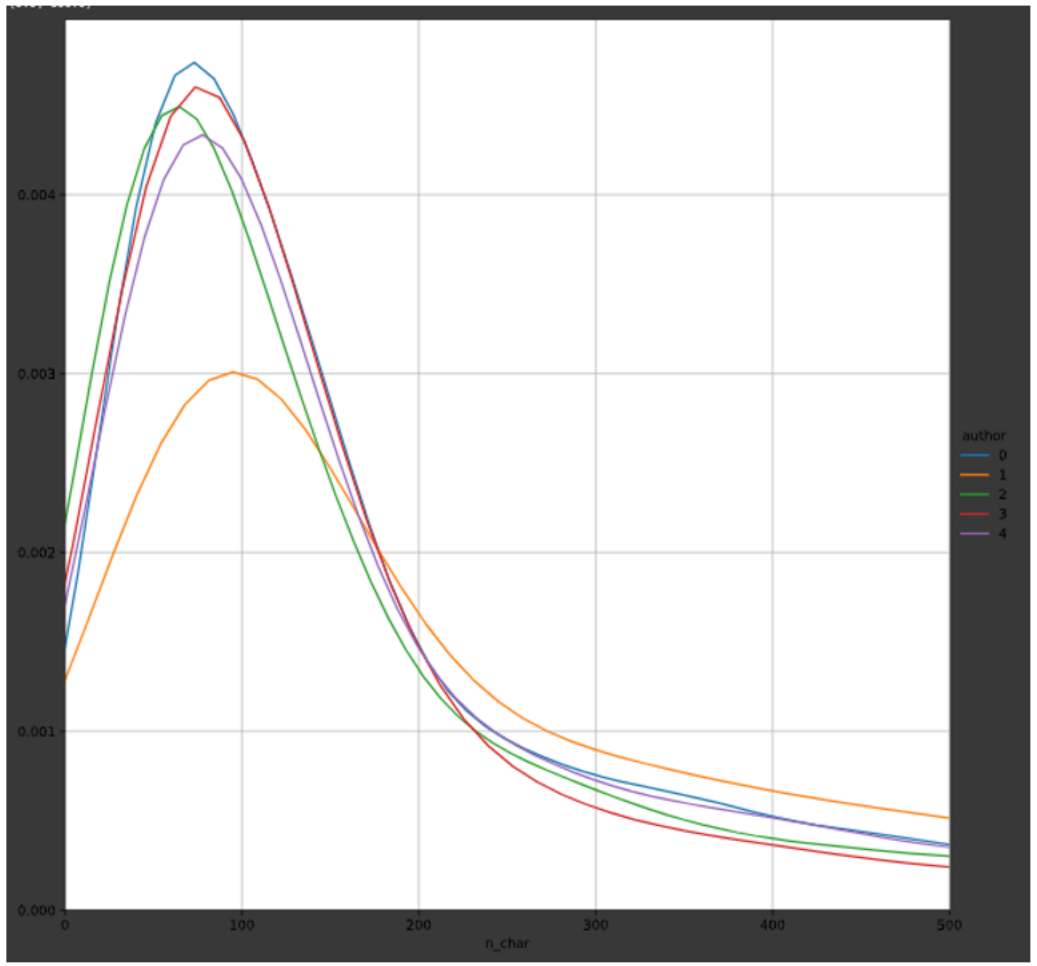
[불용어 제거 후 작가별 최빈 단어 Top 5]

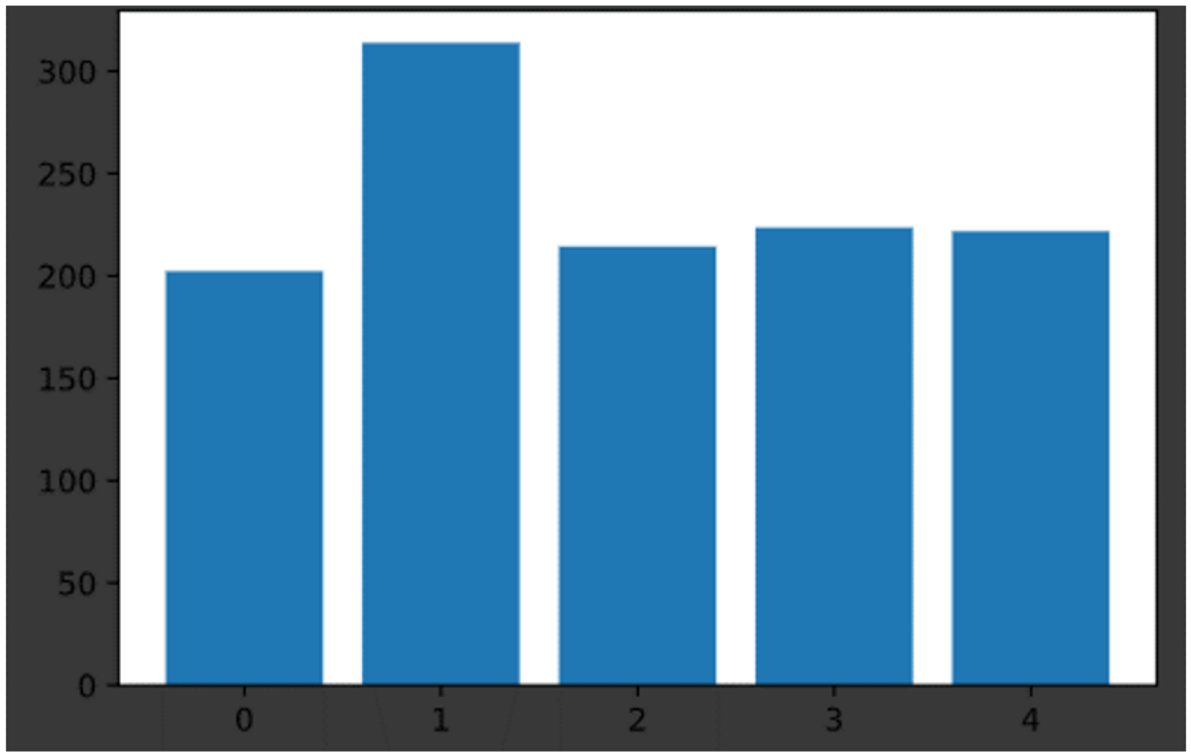


NLP - EDA

>> **Q. 작가별로 쓰는 문장의 길이는 다를 것이다.**

[작가별 소설의 길이 데이터 분석]



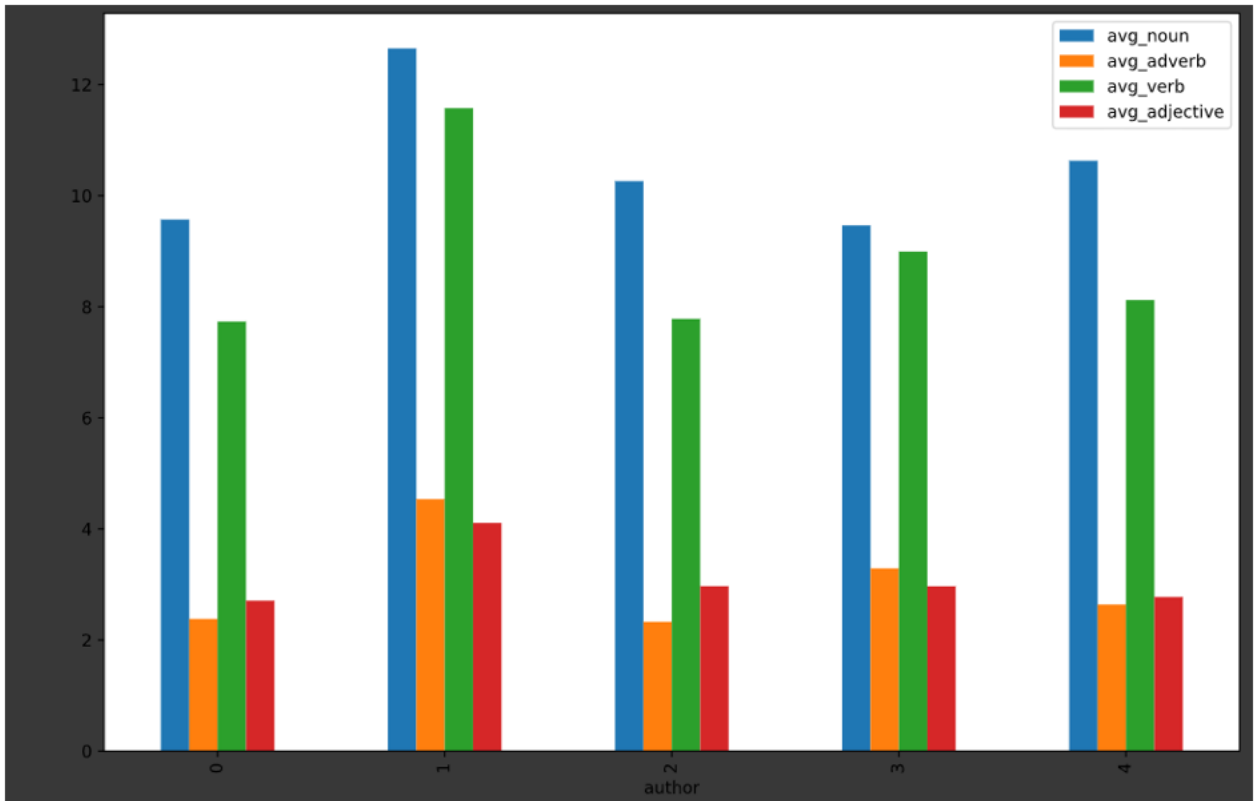


"분포에 차이가 보이지 않아 특징으로 쓰기에 부적합"

NLP - EDA

>> **Q. 작가별로 문장 품사에 구성이 다를 것이다.**

[작가별 문장속 품사 데이터 분석]

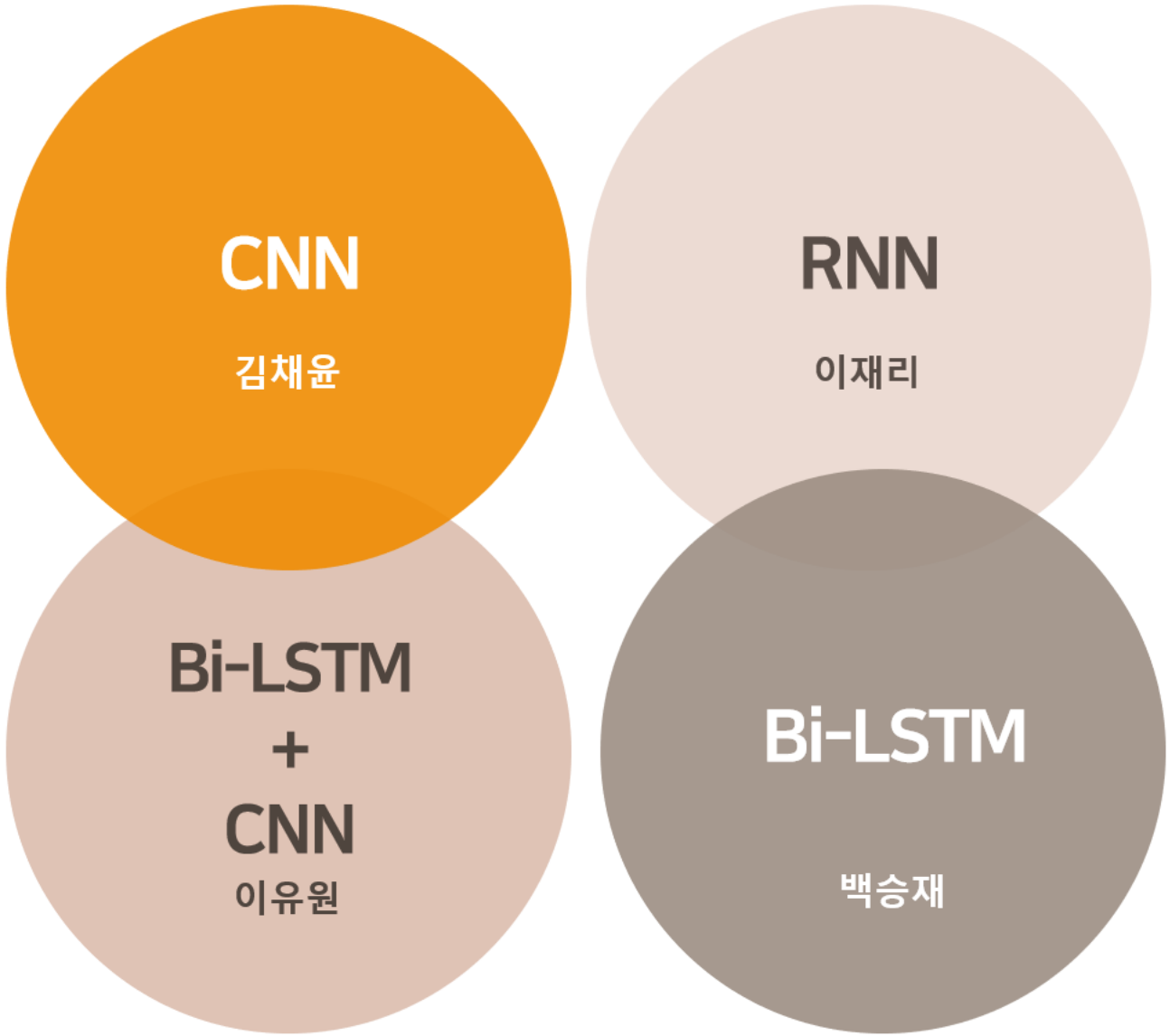


평균적으로 1번 작가가 미사여구를 많이 쓴다고 볼 수 있다.

	text_proc	pos_tags
He wa almost choking . There wa so much , so m...	PRP VBZ RB NN . EX VBZ RB JJ , RB RB PRP VBD T...
" Your sister asked for it , I suppose ? "	VB PRP$ NN VBD IN PRP , PRP VBP . NN
She wa engaged one day a she walked , in perus...	PRP VBD JJ CD NN DT PRP VBD , IN VBG NNP NNP V...
The captain wa in the porch , keeping himself ...	DT NN NN IN DT NN , VBG PRP RB IN IN DT NN IN ...
" Have mercy , gentleman ! " odin flung up his...	NNS VBP VBN , NN . JJ NN VBD RP PRP$ NN . JJ N...

author	noun	adverb	verb	adjective	count	avg_noun	avg_adverb	avg_verb	avg_adjective
0	126651	31484	102361	35830	13235	9.569399	2.378844	7.734114	2.707216
1	91359	32757	83588	29657	7222	12.650097	4.535724	11.574079	4.106480
2	118567	26891	89912	34305	11554	10.261987	2.327419	7.781894	2.969102
3	142577	49555	135452	44695	15063	9.465379	3.289849	8.992365	2.967204
4	82953	20589	63377	21653	7805	10.628187	2.637924	8.120051	2.774247

Modeling Overview



CNN Overview

001 POS-Tags + CNN

002 (Words and POS-Tags) + CNN

CNN

001 POS-Tags + CNN

Step 1. 문장 속 품사 분석

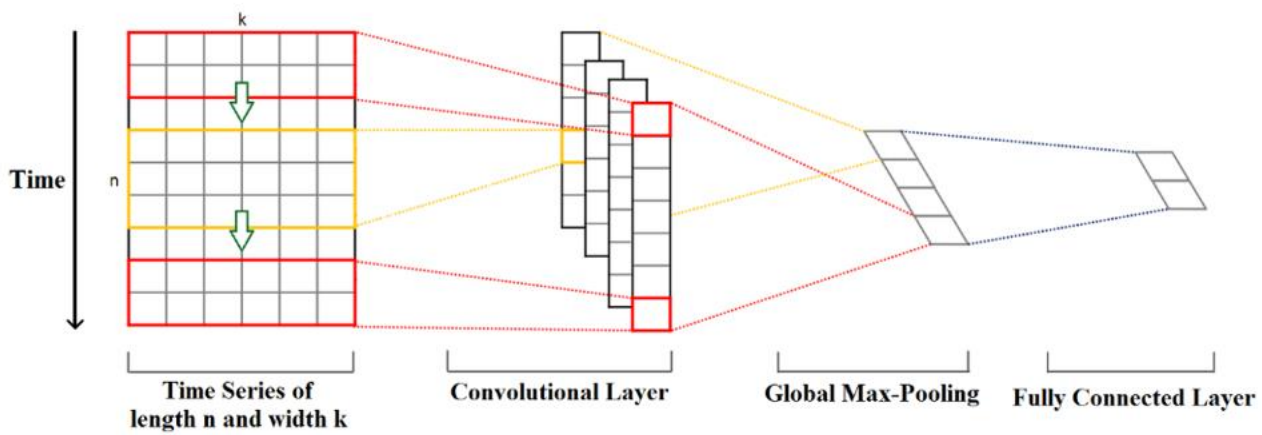
Step 2. Keras tokenizer
(texts_to_sequences)

각 텍스트를
정수 시퀀스로 변환

Step 3. 1D CNN 학습

text_proc	pos_tags
He wa almost choking . There wa so much , so m...	PRP VBZ RB NN . EX VBZ RB JJ , RB RB PRP VBD T...
" Your sister asked for it , I suppose ? "	VB PRP$ NN VBD IN PRP , PRP VBP . NN
She wa engaged one day a she walked , in perus...	PRP VBD JJ CD NN DT PRP VBD , IN VBG NNP NNP V...
The captain wa in the porch , keeping himself ...	DT NN NN IN DT NN , VBG PRP RB IN IN DT NN IN ...
" Have mercy , gentleman ! " odin flung up his...	NNS VBP VBN , NN . JJ NN VBD RP PRP$ NN . JJ N...

[data processing]



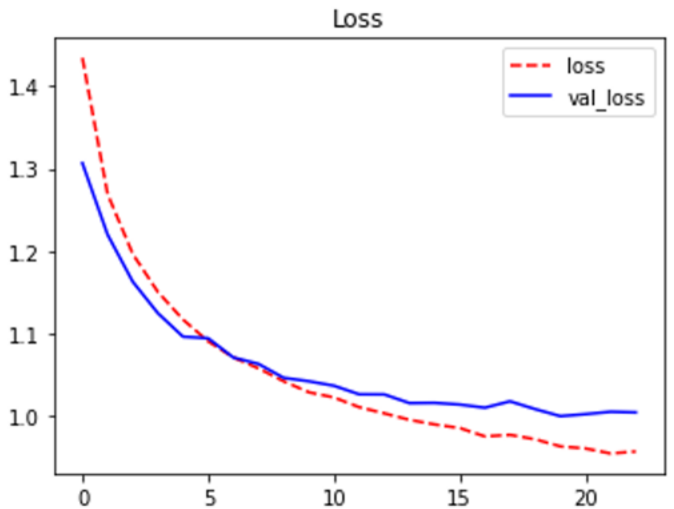
[1D CNN]

CNN

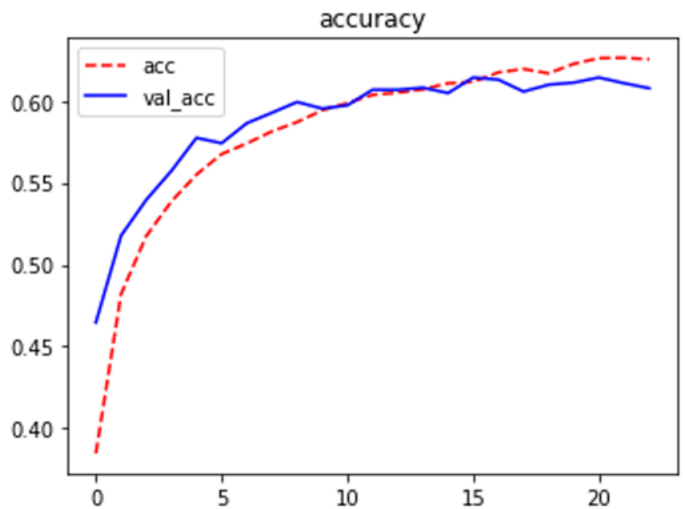
001 POS-Tags + CNN

```
Model: "sequential_91"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_91 (Embedding)     (None, 200, 200)          9200
_________________________________________________________________
dropout_169 (Dropout)        (None, 200, 200)          0
_________________________________________________________________
conv1d_194 (Conv1D)          (None, 196, 200)          200200
_________________________________________________________________
global_max_pooling1d_107 (Gl (None, 200)               0
_________________________________________________________________
dropout_170 (Dropout)        (None, 200)               0
_________________________________________________________________
dense_93 (Dense)             (None, 5)                 1005
=================================================================
Total params: 210,405
Trainable params: 210,405
Non-trainable params: 0
_________________________________________________________________
None
```

CNN Model Summary




```
loss:     0.9554
val_loss: 1.0040
```




```
accuracy:     0.6253
val_accuracy: 0.6077
```

POS-Tags + CNN Result

002

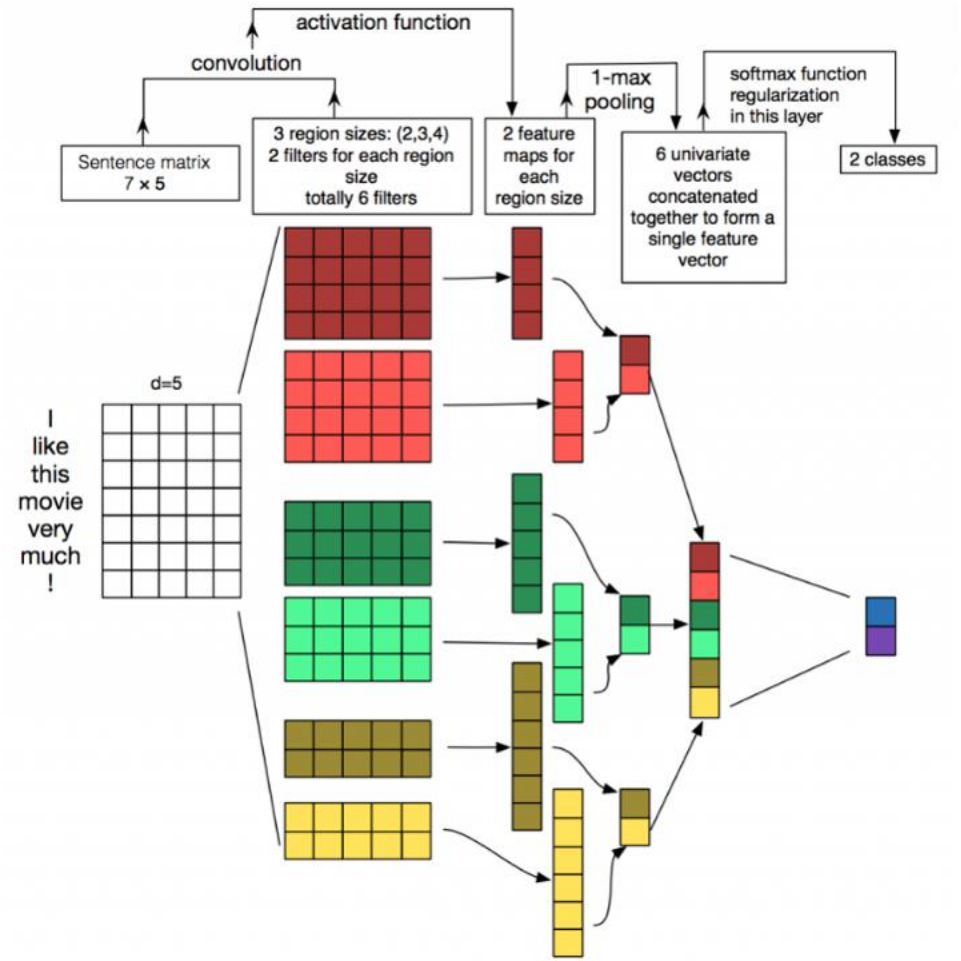
(Words and POS-Tags) + CNN

Step 1. 문장 속 각 단어에 품사 tagging

Step 2. 문장 형태소 단위로 분리

Step 3. Keras tokenizer (texts_to_sequences)

Step 4. 1D CNN 학습

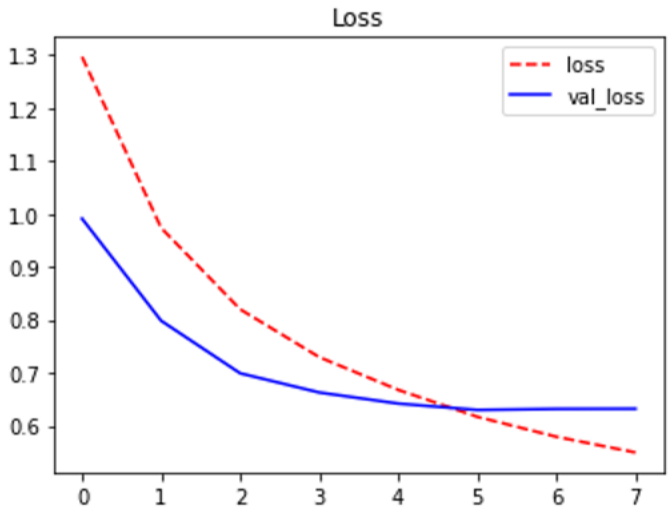


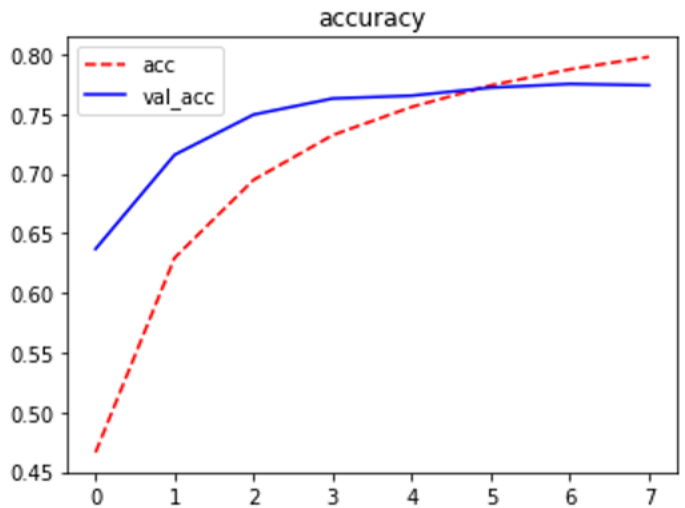
출처 : Zhang, Y., & Wallace, B. (2015). "Convolutional Neural Networks for Sentence Classification",

CNN

002 (Words and POS-Tags) + CNN

```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
pos_input (InputLayer)          [(None, 529)]        0
__________________________________________________________________________________________________
text_input (InputLayer)         [(None, 529)]        0
__________________________________________________________________________________________________
embedding_8 (Embedding)         (None, 529, 10)      460         pos_input[0][0]
__________________________________________________________________________________________________
embedding_9 (Embedding)         (None, 529, 10)      448070      text_input[0][0]
__________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 529, 20)      0           embedding_8[0][0]
                                                                 embedding_9[0][0]
__________________________________________________________________________________________________
conv1d_16 (Conv1D)              (None, 529, 20)      820         concatenate_8[0][0]
__________________________________________________________________________________________________
conv1d_17 (Conv1D)              (None, 529, 20)      820         concatenate_8[0][0]
__________________________________________________________________________________________________
conv1d_18 (Conv1D)              (None, 529, 20)      820         concatenate_8[0][0]
__________________________________________________________________________________________________
conv1d_19 (Conv1D)              (None, 529, 20)      820         concatenate_8[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_16 (Global (None, 20)           0           conv1d_16[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_17 (Global (None, 20)           0           conv1d_17[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_18 (Global (None, 20)           0           conv1d_18[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_19 (Global (None, 20)           0           conv1d_19[0][0]
__________________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, 80)           0           global_max_pooling1d_16[0][0]
                                                                 global_max_pooling1d_17[0][0]
                                                                 global_max_pooling1d_18[0][0]
                                                                 global_max_pooling1d_19[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 80)           0           concatenate_9[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 5)            405         dropout_4[0][0]
==================================================================================================
Total params: 452,215
Trainable params: 452,215
Non-trainable params: 0
```






```
loss:     0.5492
val_loss: 0.6319
```

```
accuracy:     0.7978
val_accuracy: 0.7741
```

Words & POS-Tags
+ CNN
Result

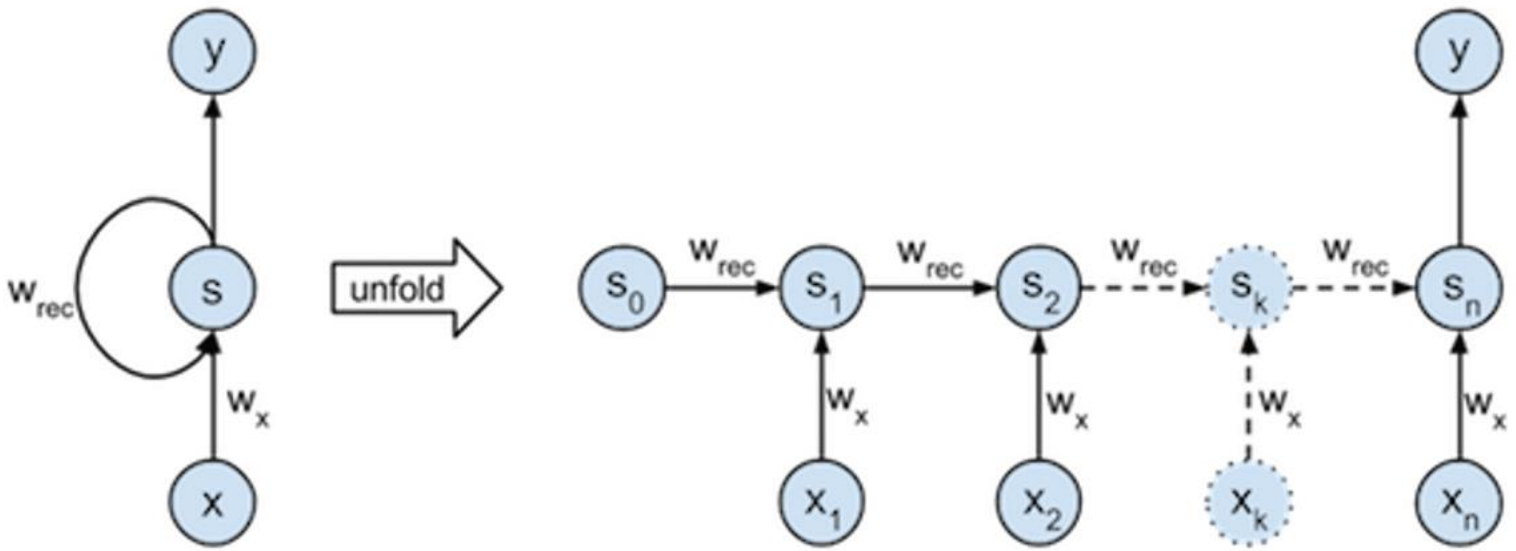
simpleRNN Overview

Step 1. 문장 부호 제거

Step 2. 불용어 제거

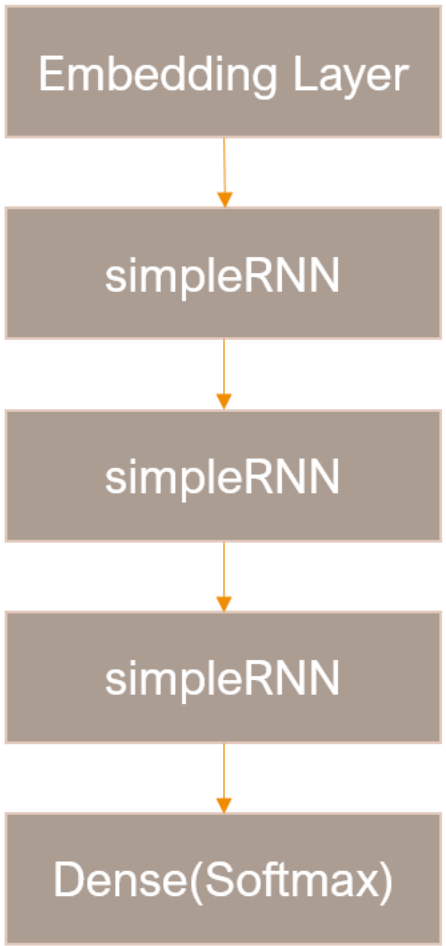
Step 3. 토큰화 및 원-핫 벡터

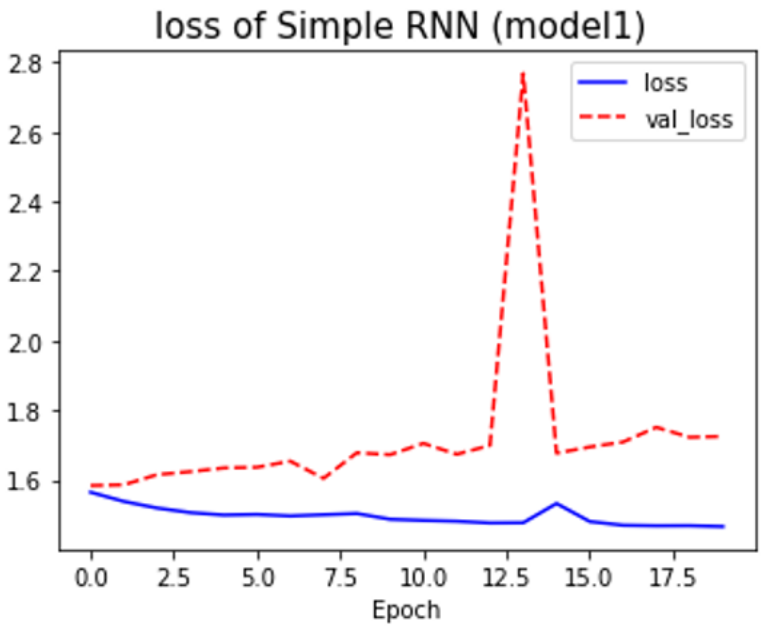
Step 4. Simple RNN



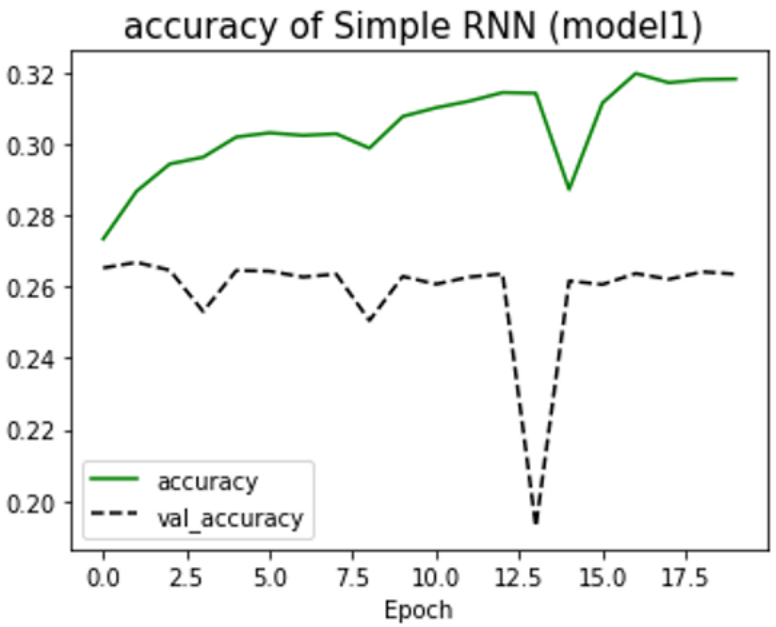
[simple RNN]

simpleRNN Result



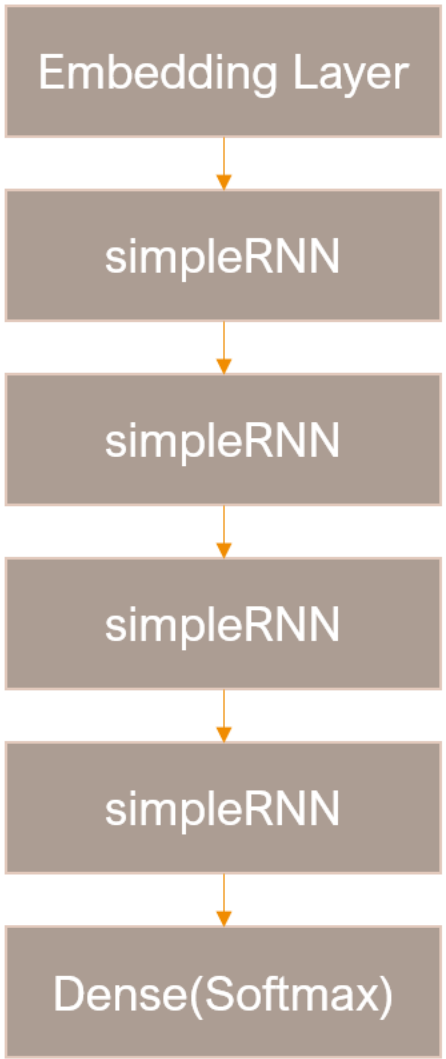


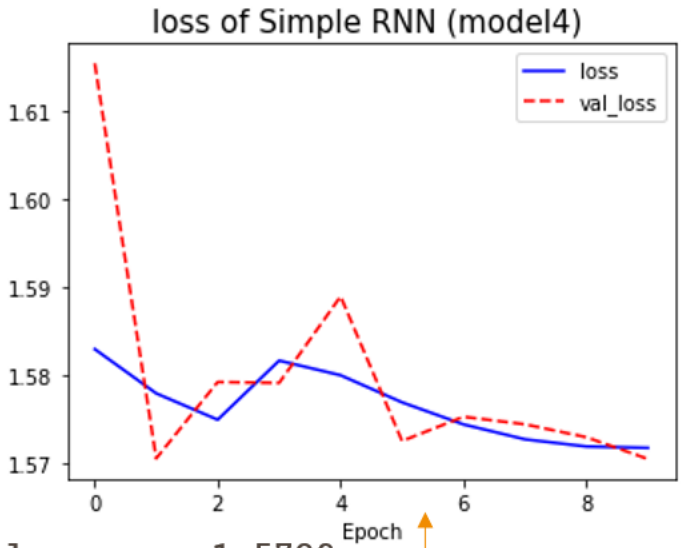

```
loss:     1.4663
val_loss: 1.7253
```



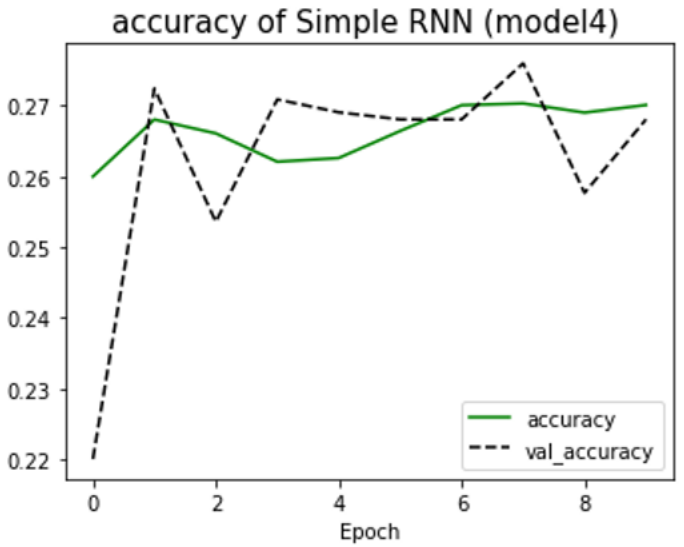

```
accuracy: 0.3183
val_accuracy: 0.2636
```

simpleRNN Result



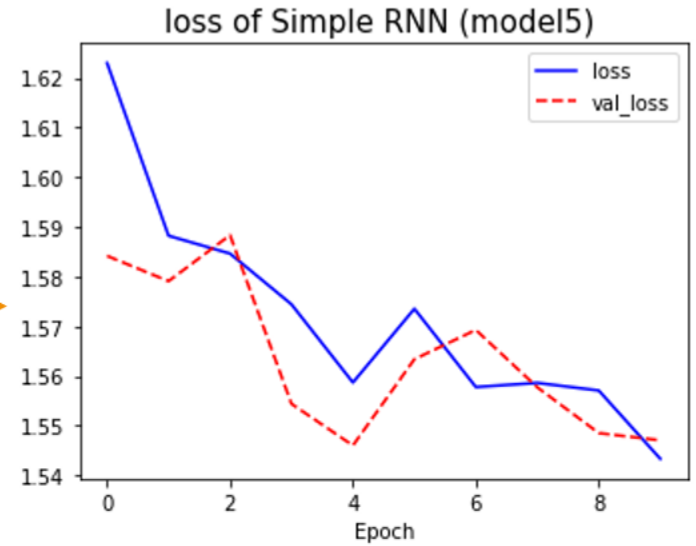



```
loss:     1.5729
val_loss: 1.5706
```

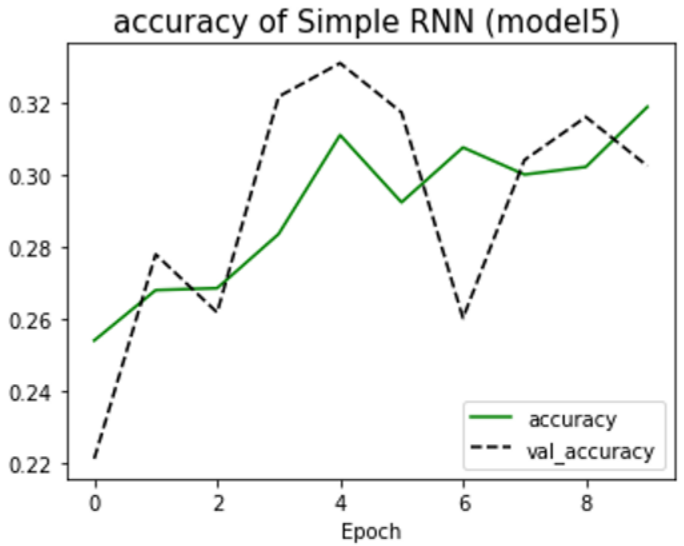



```
accuracy:     0.2688
val_accuracy: 0.2680
```

```
acc:     0.3151
val_acc: 0.3023
```



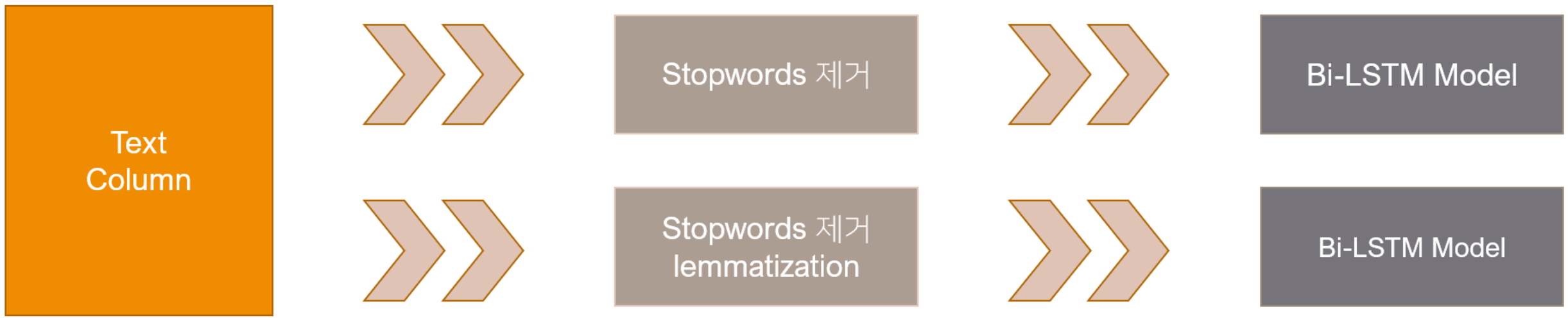

```
loss:     1.5457
val_loss: 1.5471
```



Bi-LSTM Overview

001 Stopwords 제거 + Bi-LSTM

002 (Stopwords 제거 & lemmatization) + Bi-LSTM



Bi-LSTM

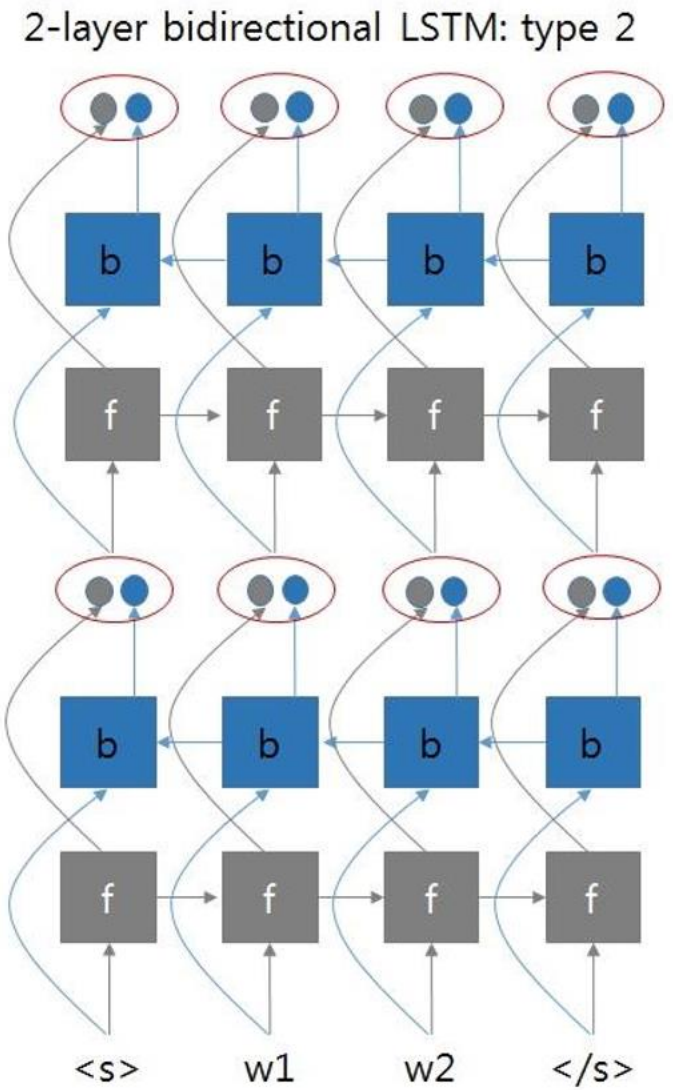
Step 1. 데이터 전처리
(불용어 제거+ 표제어 추출)

Step 2. 단어 벡터화 + 패딩

Step 3. 데이터 split test, valid

Step 4. Word2vec 임베딩

Step 5. Bi-LSTM 모델

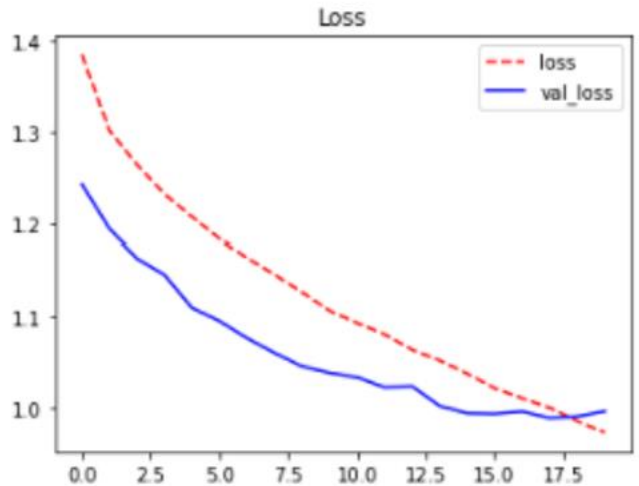


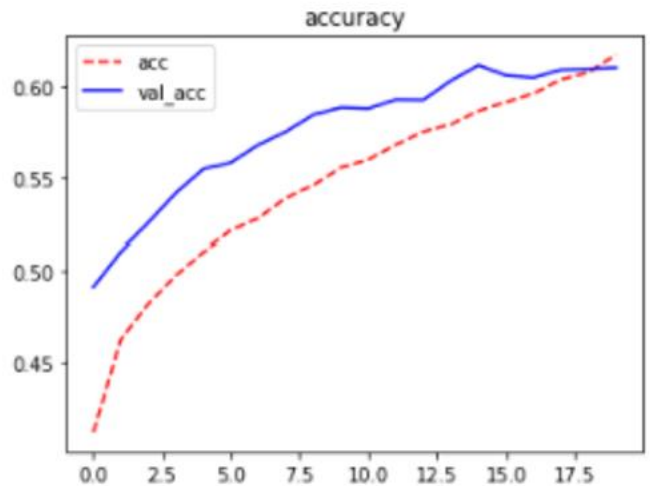
Bi-LSTM – Result

001 (Stopwords 제거 & lemmatization) + Bi-LSTM

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 64)           1654272
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 100, 64)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 100, 256)          197632
_________________________________________________________________
bidirectional_1 (Bidirection (None, 256)               394240
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 64)                16448
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 325
=================================================================
Total params: 2,262,917
Trainable params: 608,645
Non-trainable params: 1,654,272
_________________________________________________________________
```

[Bi-LSTM Summary]






```
loss:     0.9655
val_loss: 0.9963
```

```
accuracy: 0.6204
val_accuracy: 0.6105
```

[Bi-LSTM Result]

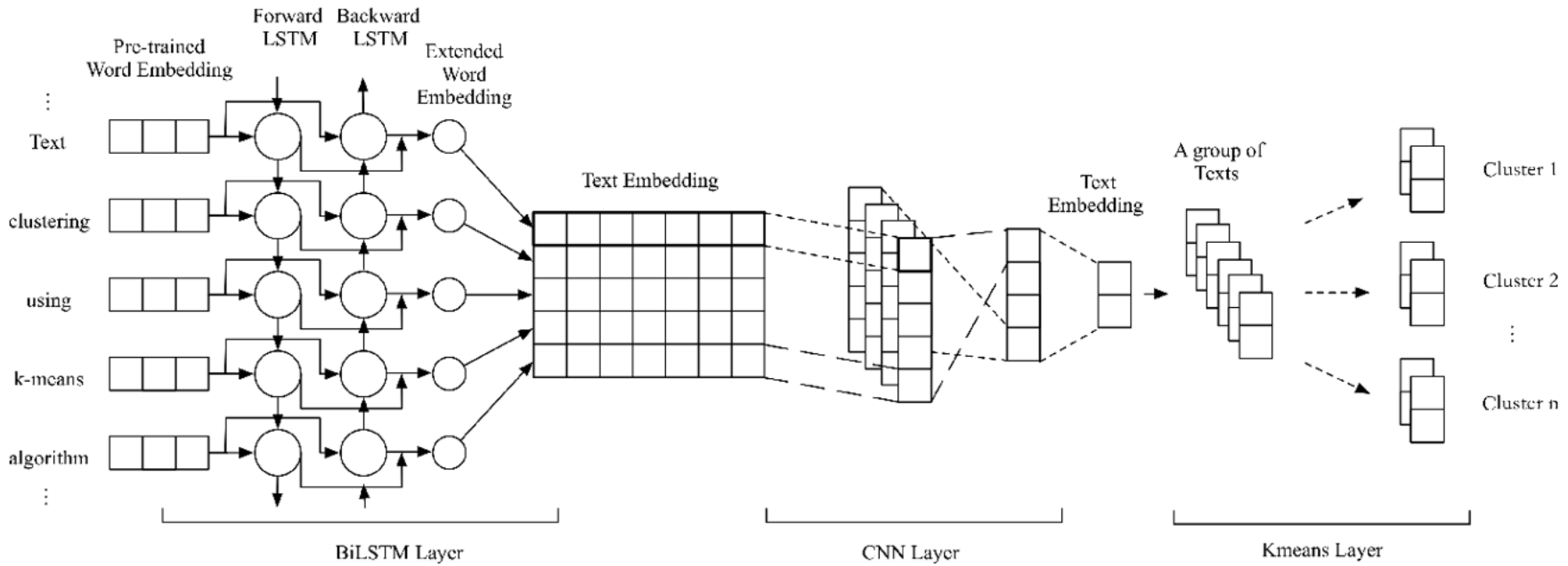
Bi-LSTM & CNN Overview

Step 1. 문장 부호 제거

Step 2. BERT Pre-trained tokenizer

Step 3. Train – LSTM_CNN

Step 4. Eavaluate

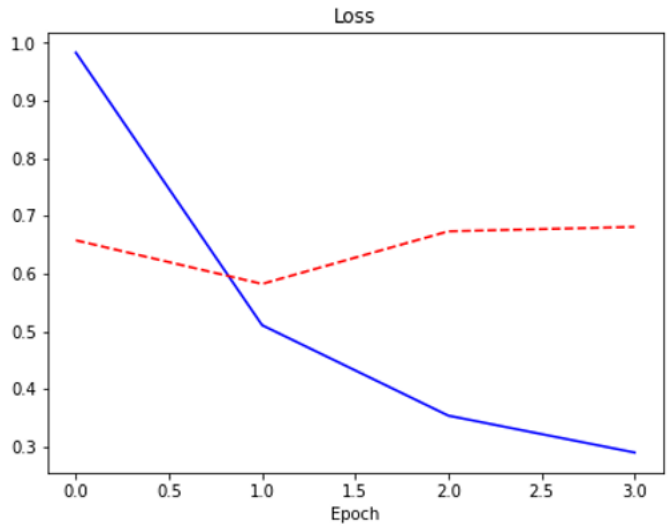


출처 :LSTM과 CNN을 이용한 텍스트 분류, 한국정보통신학회 학술집 (2019)

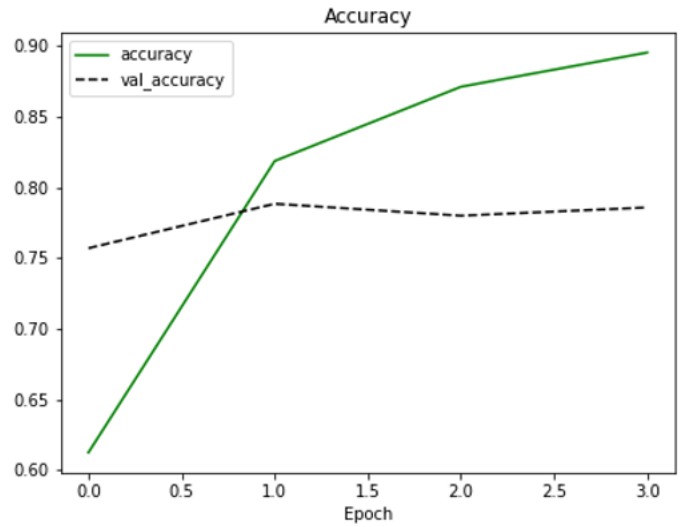
Bi-LSTM & CNN Result

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_16 (Embedding)     (None, 472, 100)          3052300
_________________________________________________________________
bidirectional_15 (Bidirectio (None, 472, 256)          234496
_________________________________________________________________
conv1d_15 (Conv1D)           (None, 472, 128)          163968
_________________________________________________________________
global_max_pooling1d_10 (Glo (None, 128)               0
_________________________________________________________________
dropout_13 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_23 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_24 (Dense)             (None, 5)                 645
=================================================================
Total params: 3,467,921
Trainable params: 3,467,921
Non-trainable params: 0
_________________________________________________________________
```

Bi-LSTM + CNN
Summary



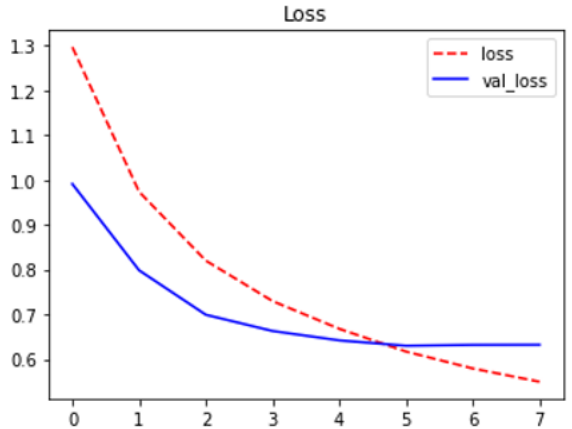

```
loss:     0.2663
val_loss: 0.6809
```

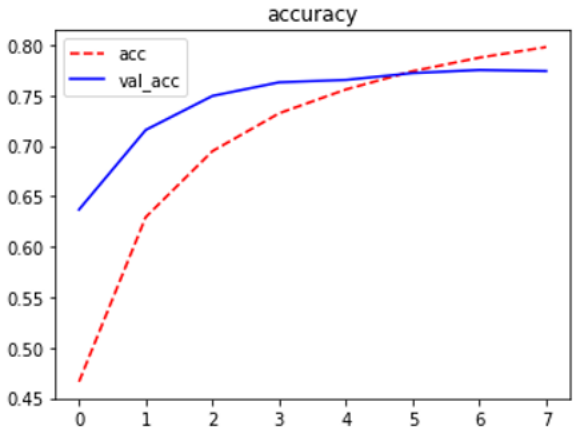



```
accuracy: 0.9043
val_accuracy: 0.7857
```

Bi-LSTM + CNN
Result

결론

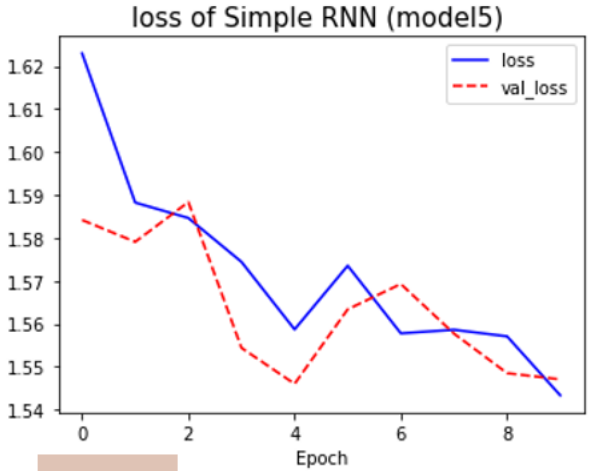


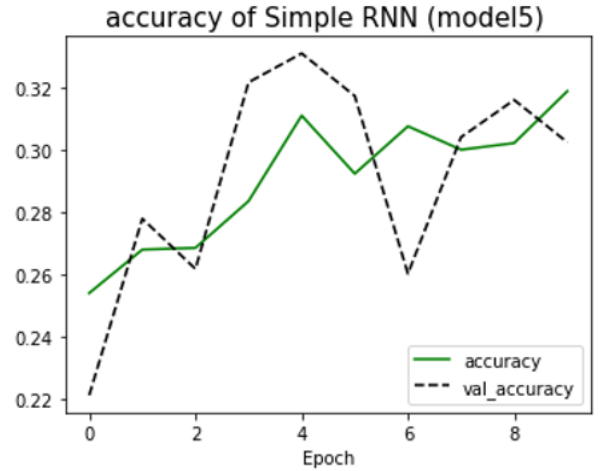


loss: 0.5492
val_loss: 0.6319

acc: 0.7978
val_acc: 0.7741

CNN





RNN

loss: 1.5457
val_loss: 1.5471

acc: 0.3151
val_acc: 0.3023

loss: 0.2663
val_loss: 0.6809

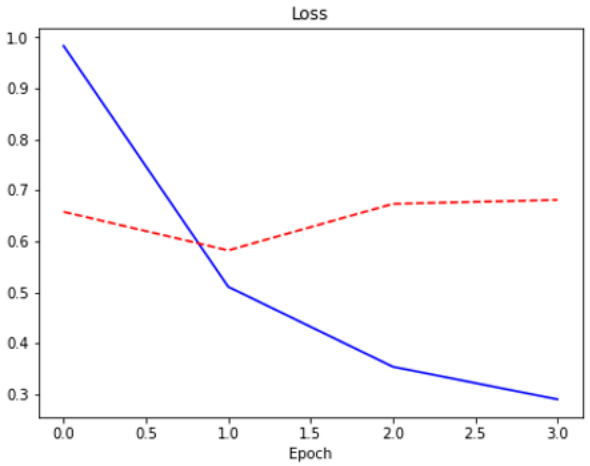
acc: 0.9043
val_acc: 0.7857

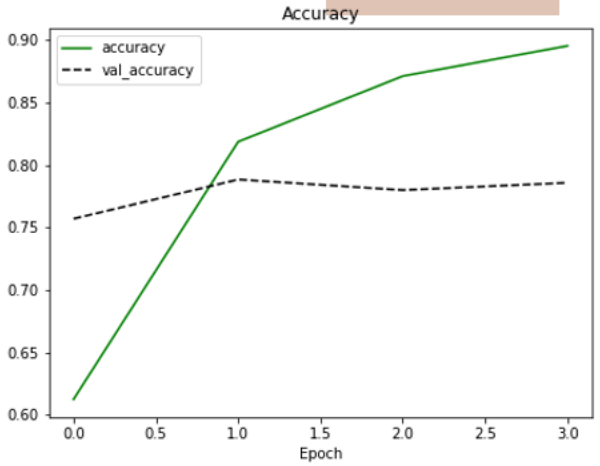
Bi-LSTM CNN

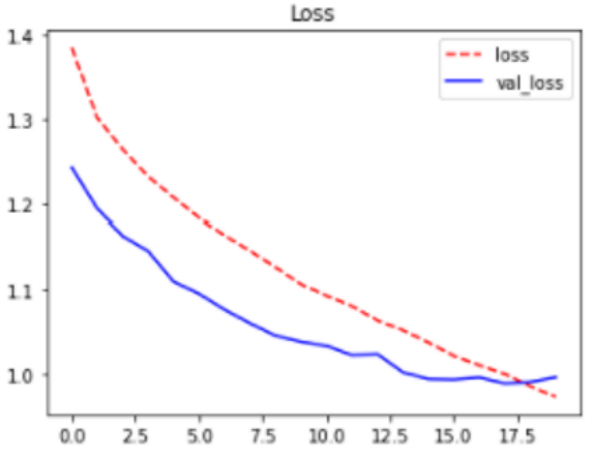
Bi-LSTM

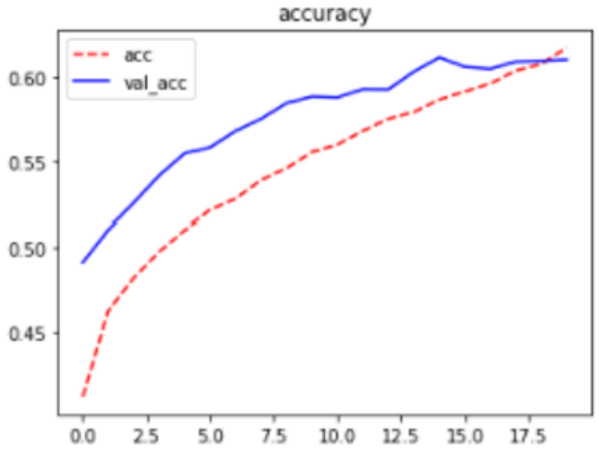
loss: 0.8775
val_loss: 1.0318

acc: 0.6548
val_acc: 0.5964









참고

[1] Johnson, R., & Zhang, T. *"Effective Use of Word Order for Text Categorization with Convolutional Neural Networks."* To Appear: NAACL-2015,

[2] Zhang, Y., & Wallace, B. A Sensitivity Analysis of (and Practitioners' Guide to) "*Convolutional Neural Networks for Sentence Classification",* (2015).

[3] Santos, C., & Zadrozny, B. *"Learning Character-level Representations for Part-of-Speech Tagging. Proceedings of the 31st International Conference on Machine Learning",* ICML-14(2011), 1818–1826.

[4] Zhang, X., Zhao, J., & LeCun, Y. (2015). *"Character-level Convolutional Networks for Text Classification",* 1–9.

[5] Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2015). *"Character-Aware Neural Language Models."*

[6] Seong-Yoon Shin, Kwang-Seong Shin, Hyun-Chang Lee, *"Text Classification Using LSTM-CNN",* The Korea Institute of Information and Communication Engineering 2019.10, 692-694(3 pages)

[7] Ho-yeon Park, Kyoung-jae Kim, *"Sentiment Analysis of Movie Review Using Integrated CNN-LSTM Model",* Korea Intelligent Information Systems Society 2019.12, 141-154(14 pages)

감사합니다