소설 작가 분류 AI

정석영, 최유리, 최디도, 엄남영, 박민춘

STEP OF PRESENTATION

- Introduction
- EDA
- Embedding
- Train
- Conclusion

Introduction

소설 내의 문장들을 분석하여, 그에 해당하는 저자를 예측

학습 효과를 높이기 위한 다양한 방법..?

- 데이터 수
  외부 영미 소설 데이터를 추가함으로써 학습 효과 상승 기대

- 임베딩 과정
  'FastText' vs 'Glove'

- 다양한 학습 모델
  CNN / LSTM / Bi-LSTM / GRU

EDA – Custom Data

Original train : (54879,3) label : 5개(0-4)
Original test : 19617

Custom train : (83220, 3) label : 5개(0-7)
Custom test : 27318

	index	text	author
0	0	It is hard to forget repulsive things. I remem...	6
1	1	It would be tedious if given in the beadle's w...	7
2	2	"Very good. Shall we argue about it here in p...	2
3	3	"What! and I as high as a tree and as big as a...	6
4	4	"Isn't it enough, Vanya?" she cried, seeing hi...	3
...	...	...	...
83215	83215	What with the birthday visitors, and what with...	7
83216	83216	It was an old rickety door and gave at once be...	2
83217	83217	"Well then you can go to the devil," said odin...	3
83218	83218	"Don't know?"	7
83219	83219	"Not go to town!" cried Mrs. Palmer, with a la...	5

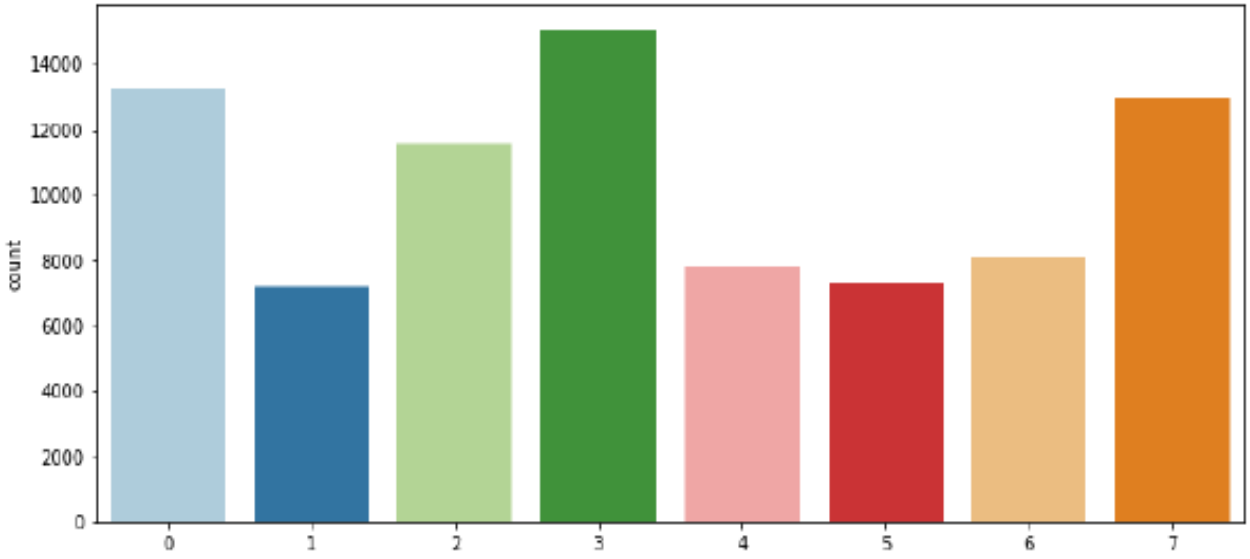
83220 rows × 3 columns

	index	text
0	0	About thirty years ago Miss Maria Ward, of Hun...
1	1	Their homes were so distant, and the circles i...
2	2	The letter was not unproductive. It re-establi...
3	3	Such were its immediate effects, and within a ...
4	4	Sir Thomas could not give so instantaneous and...
...	...	...
27313	27313	At the end of another day or two, odin growing...
27314	27314	All afternoon we sat together, mostly in silen...
27315	27315	odin, having carried his thanks to odin, proc...
27316	27316	Soon after this, upon odin's leaving the room,...
27317	27317	And all the worse for the doomed man, that the...

27318 rows × 2 columns

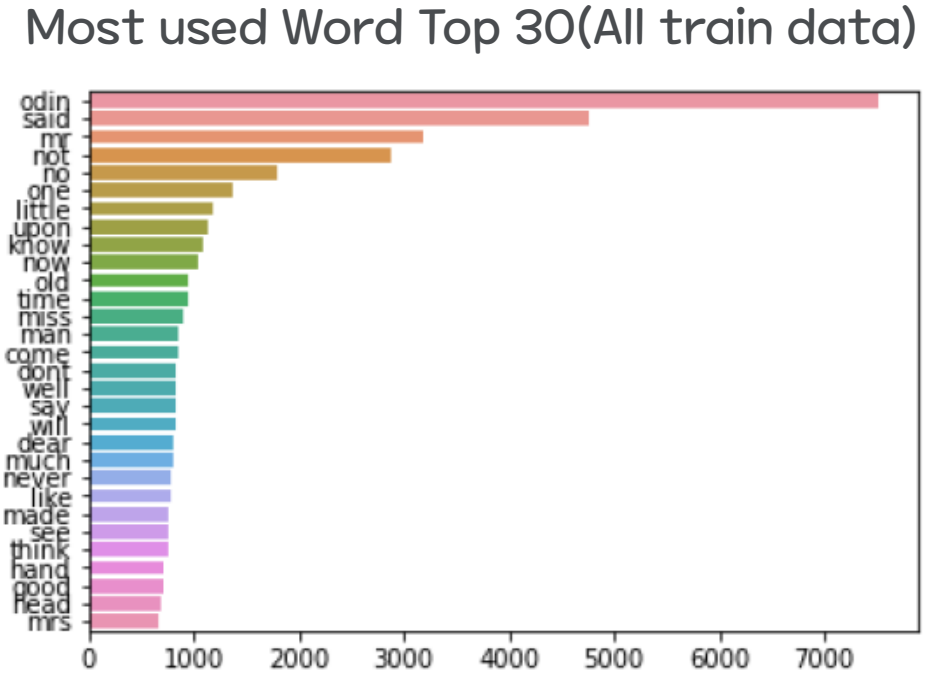
EDA – Text count

Text Count by Author



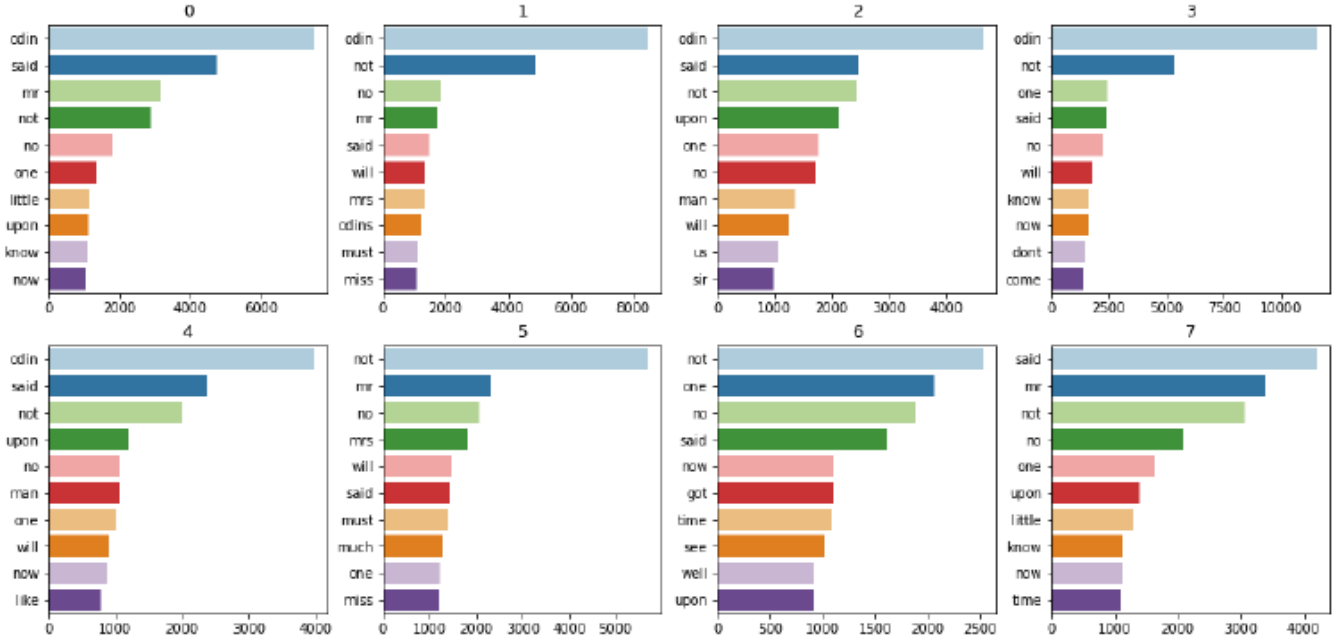
author	index
0	13235
1	7222
2	11554
3	15063
4	7805
5	7307
6	8051
7	12993

EDA - Words


Most used Word Top 30(All train data)
odin
said
mr
not
no
one
little
upon
know
now
old
time
miss
man
come
dont
well
say
will
dear
much
never
like
made
see
think
hand
good
Head
mrs
0
1000
2000
3000
4000
5000
6000
7000

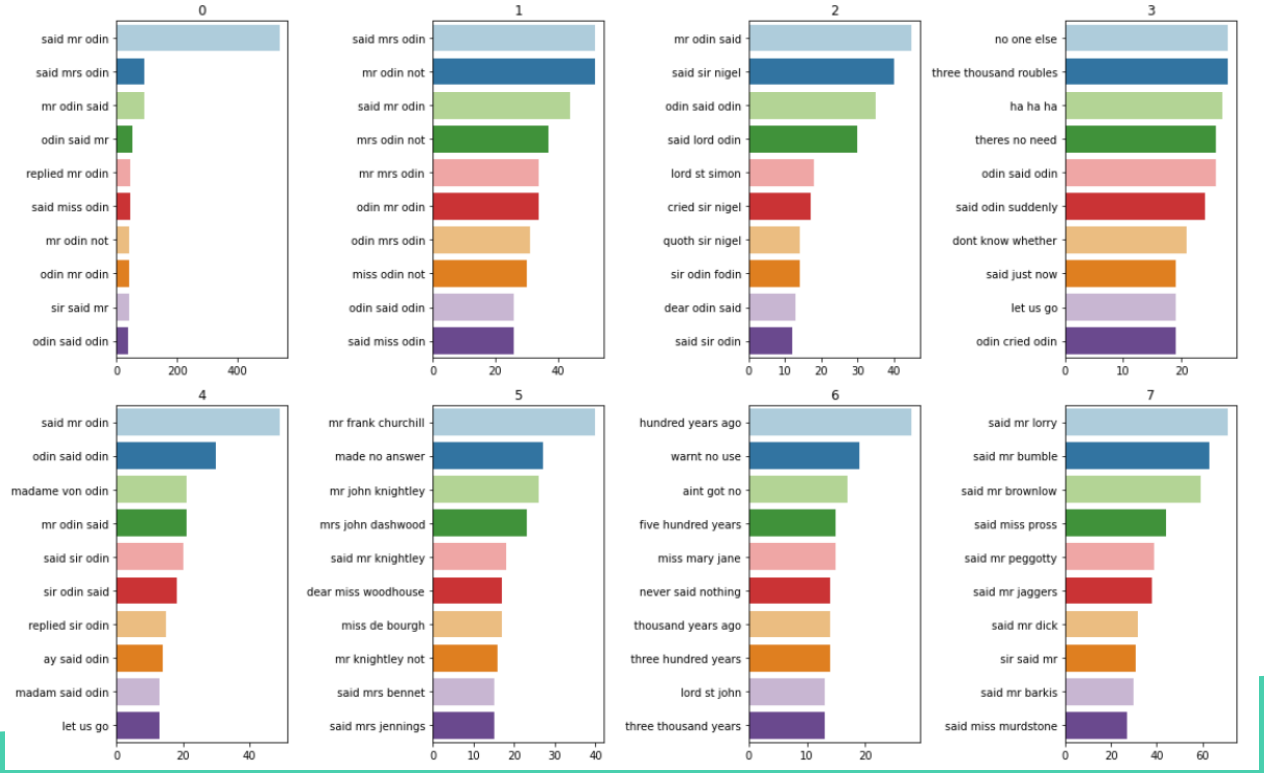
EDA - Words

Most used Word Top 10 by Author



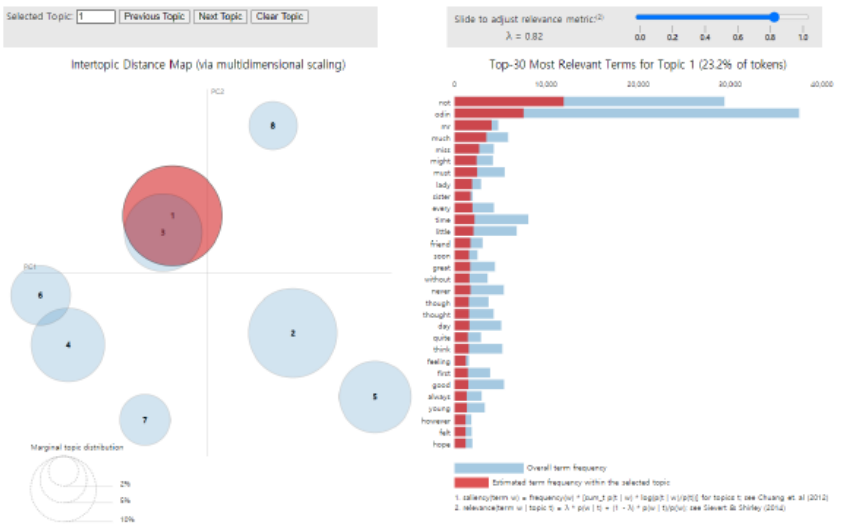
EDA - Words

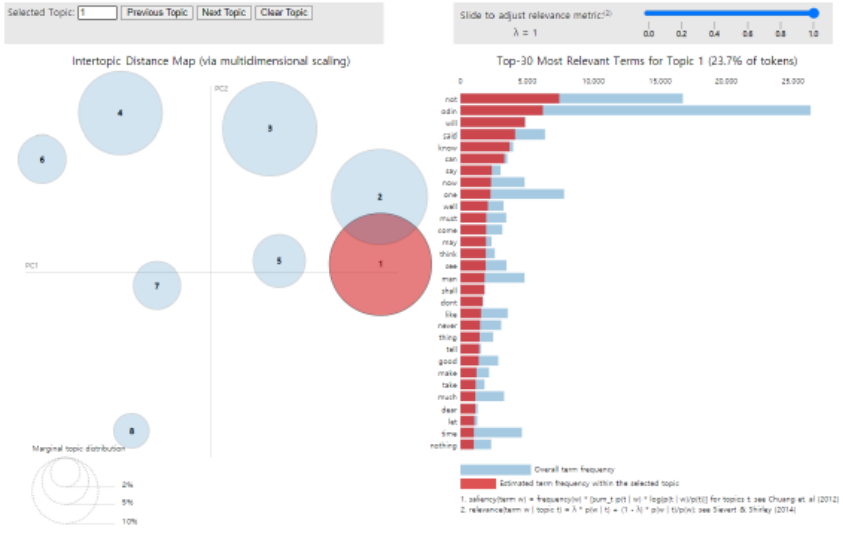
Most used 3-gram Words Top 10 by Author



EDA - Context

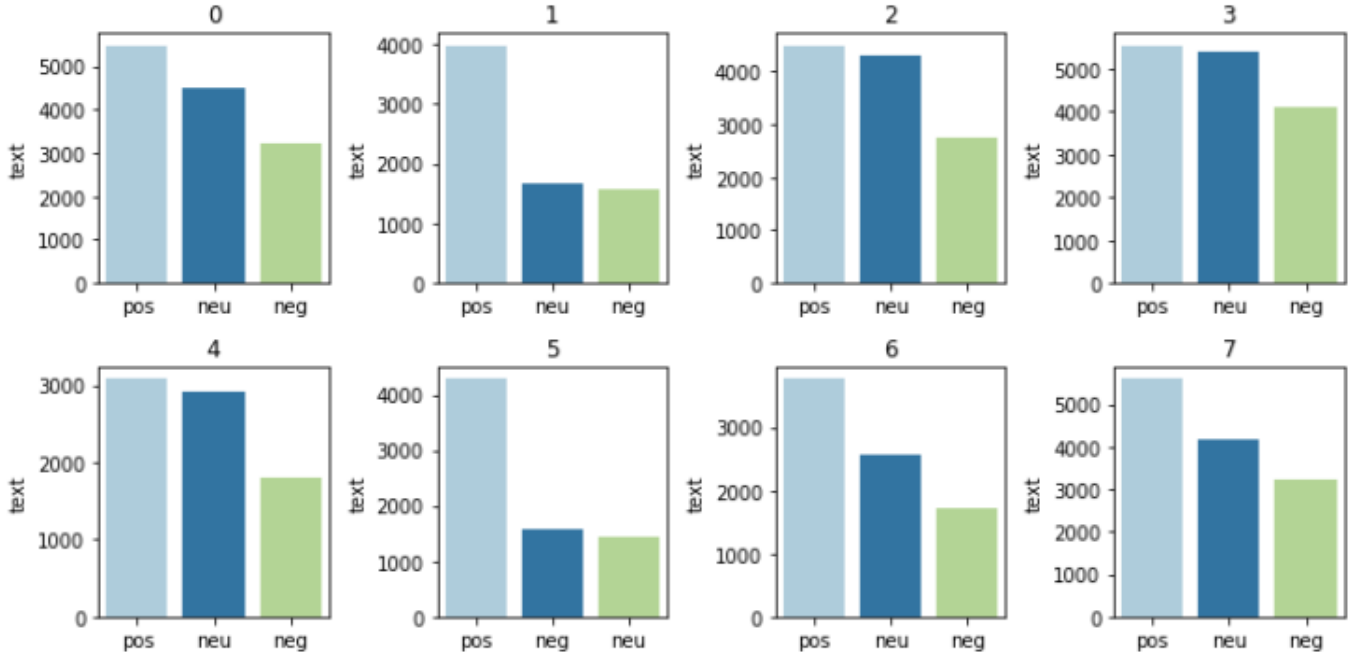
Context Similarity Distribution Plot



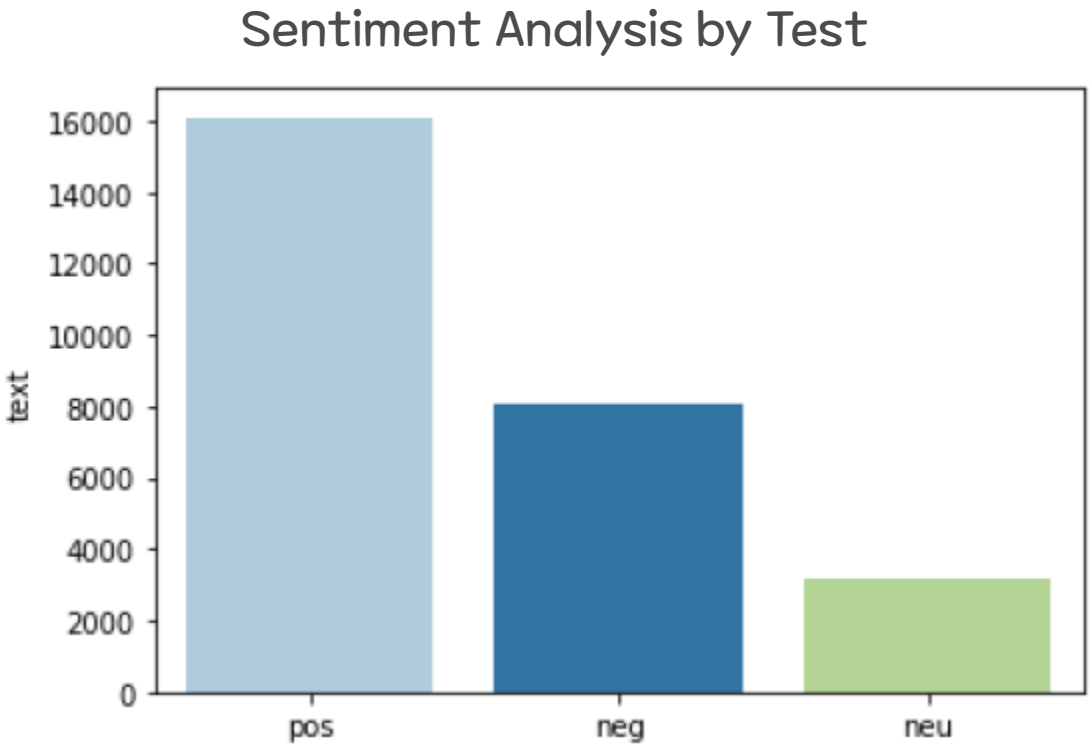


EDA - Sentiment Analysis

Sentiment Analysis by Author



EDA – Sentiment Analysis



Embedding - Glove

Glove Embedding : 스탠포드에서 개발한 NLP 임베딩 기법

- 예측 기반 기법의 Word2Vec

  실제값과 예측값에 대한 오차를 손실 함수를 통해 줄여나가며 학습
  단어 간 유추 작업에서는 상당한 성능을 보이지만, 임베딩 벡터의 한정된 크기에 의해 전체 통계 흐름은 반영 불가

- 카운트 기반 기법의 LSA

  단어의 빈도 수를 전체 통계 정보로 입력 받아, 차원을 축소시키며 의미를 추출
  전체 통계 정보를 고려하지만, 유사한 단어를 유추하는 성능은 떨어짐

위의 장점을 취하고 단점을 보완하기 위해, 윈도우 기반 동시 등장 행렬과 동시 등장 확률 메커니즘을 활용

Embedding - FastText

FastText Embedding : Facebook에서 개발한 NLP 임베딩 기법

- Word2Vec과 마찬가지로 단어의 의미를 벡터화 하여 단어 간 유사도를 반영

  Word2Vec의 단어는 하나의 단위로 취급한다면, FastText는 하나의 단어 내의 subword를 고려

- Subword

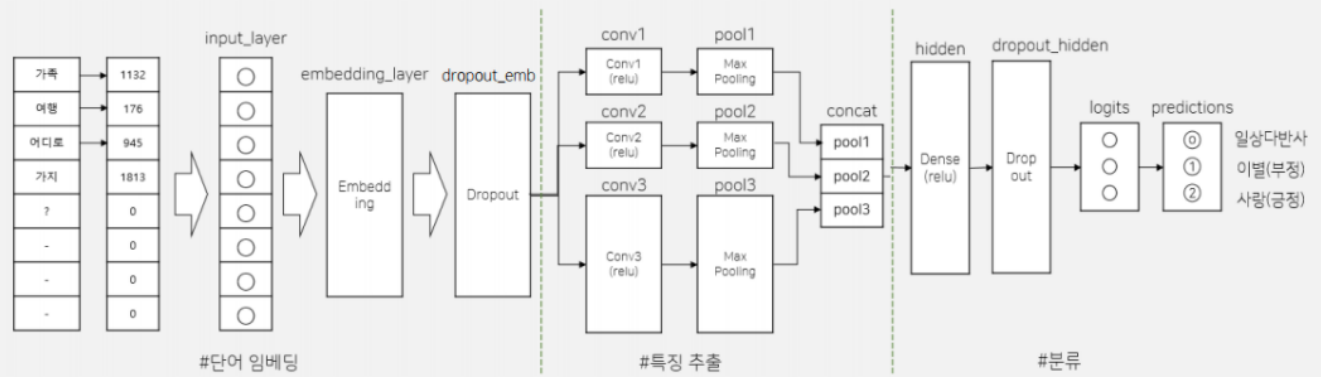
  단어를 n-gram의 구성으로 취급해, 각 subword token을 벡터화

  이를 통해 등장 빈도 수가 적은 단어 및 비정형 텍스트 데이터에 대해 Word2Vec에 비해 좋은 성능

Train – CNN

CNN 학습 전략

- 데이터 셋 확장에 따른 학습 효과 비교
- 기본 자연어 CNN



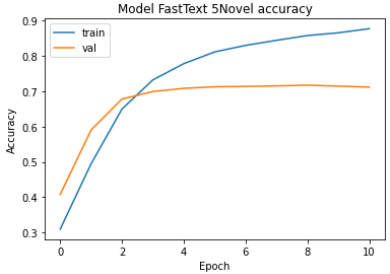
- 이를 토대로 다른 방식(Long & Wide)의 CNN을 구성해 학습 효과 비교

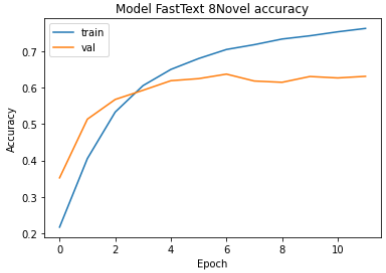
  Long Layer : Kernel size 7 – 6 – 5 – 4 – 3

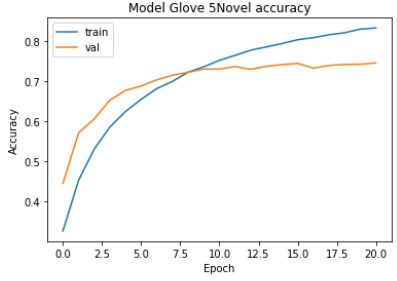
  Wide Layer : Kernel size 4 * 2 – 3 * 2 – 2 * 2

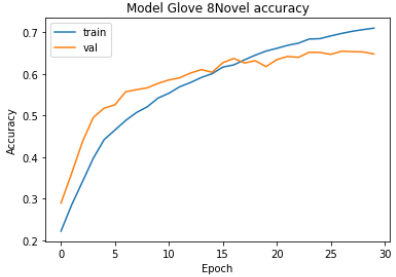
Train – CNN

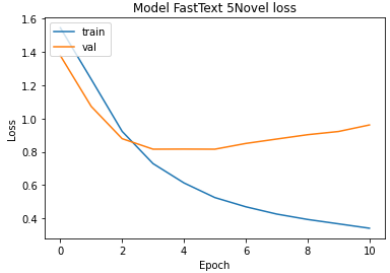
기본 CNN 모델을 활용한 학습 결과

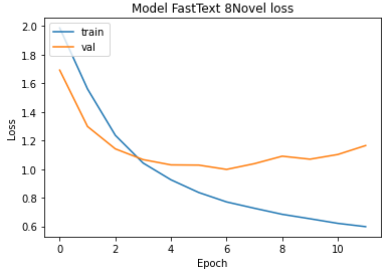


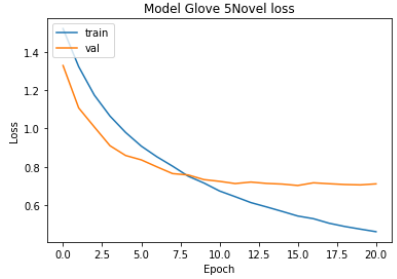


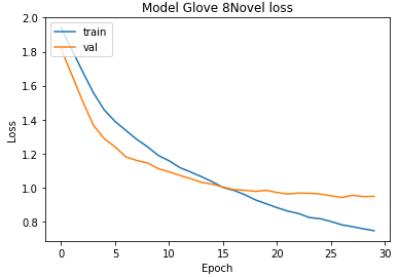






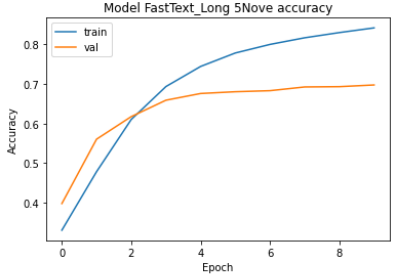


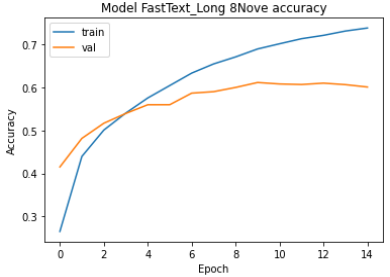


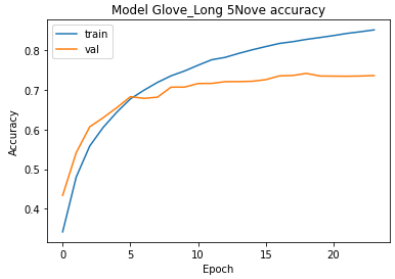


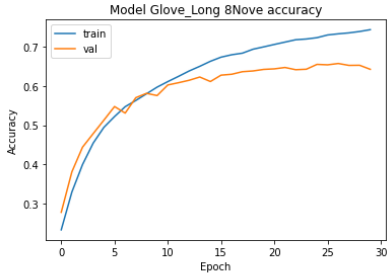
Train – CNN

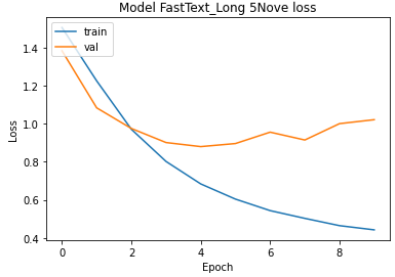
Long Layer CNN 모델을 활용한 학습 결과

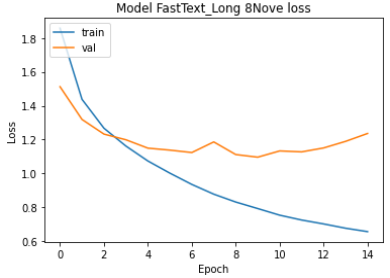


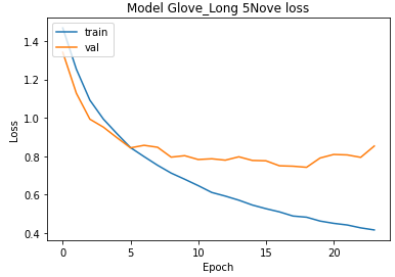


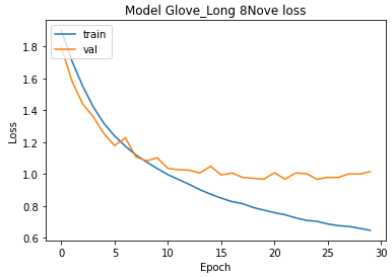






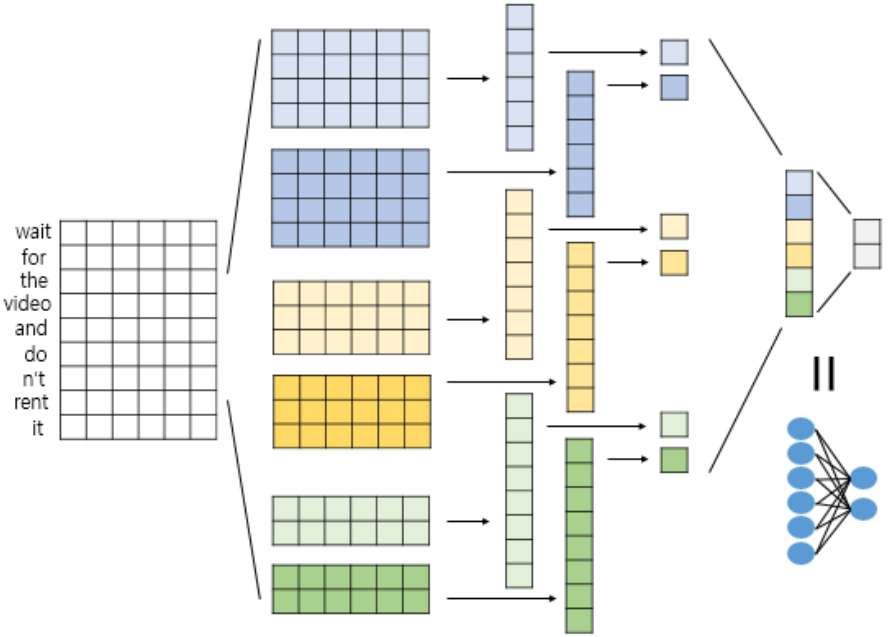






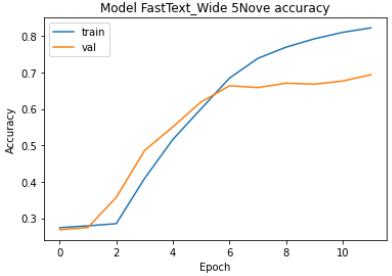
Train – CNN

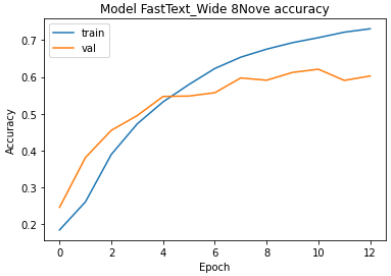
Wide Layer CNN

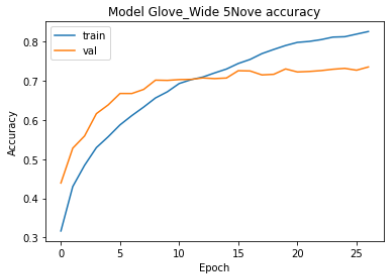


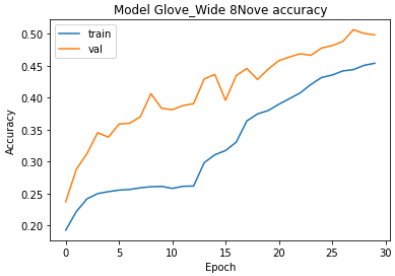
Train – CNN

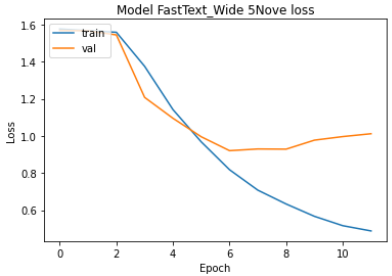
Wide Layer CNN 모델을 활용한 학습 결과

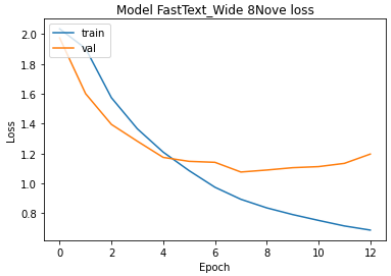


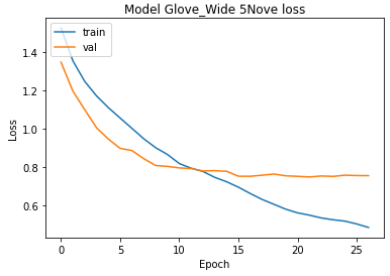


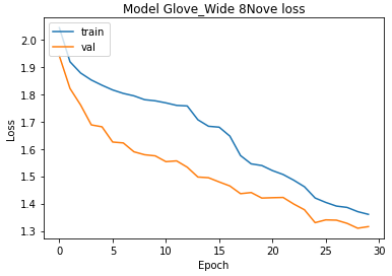






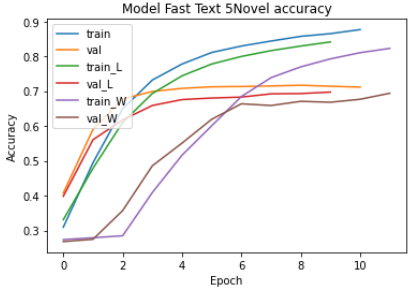


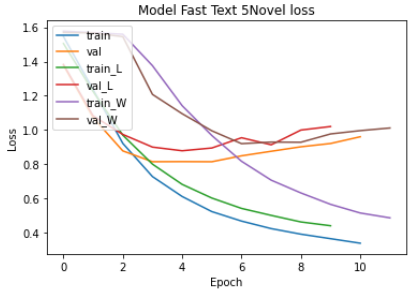




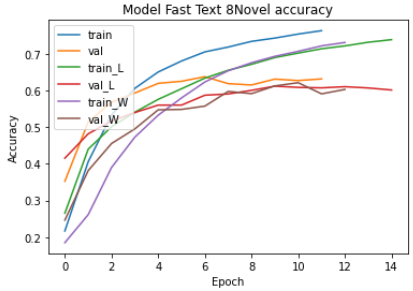
Train – CNN

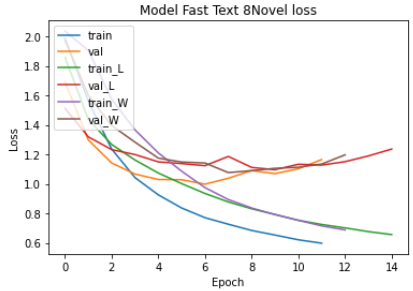
전체 CNN 모델 비교 by FastText





	Val_loss	Val_acc
Base	0.9611	0.7116
Long	1.0208	0.6973
Wide	1.0121	0.6940

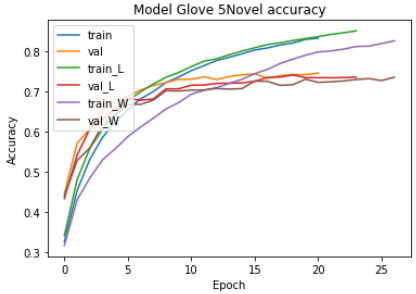


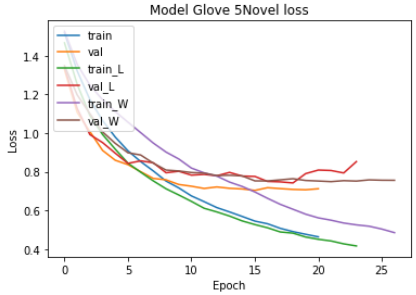


	Val_loss	Val_acc
Base	1.1648	0.6312
Long	1.2359	0.6010
Wide	1.1966	0.6025

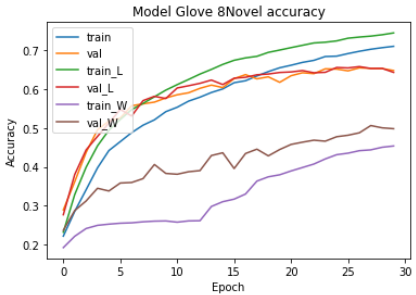
Train – CNN

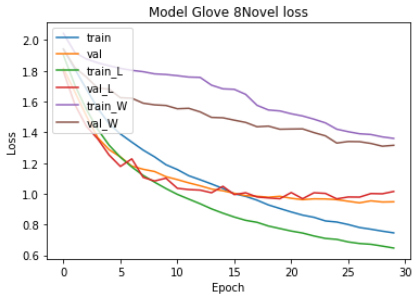
전체 CNN 모델 비교 by Glove





	Val_loss	Val_acc
Base	0.7119	0.7456
Long	0.8526	0.7358
Wide	0.7557	0.7356





	Val_loss	Val_acc
Base	0.9482	0.6480
Long	1.0153	0.6432
Wide	1.3462	0.4640

Train

다양한 모델에 따른 학습 전략

- LSTM

  Unit에 따른 LSTM 비교 학습

  LSTM 과 CNN 을 결합한 모델 학습

- Bidirectional LSTM

  2개 층 Bi-LSTM 과 1개 층 + Attention Mechanism 비교 학습

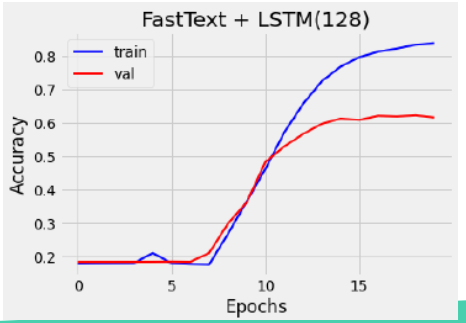
- GRU

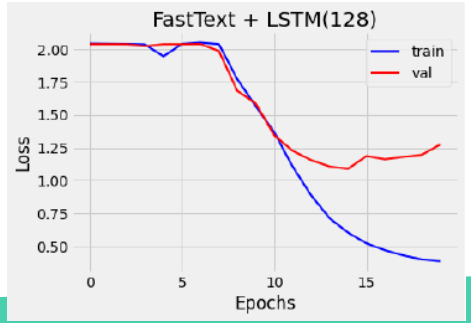
  단방향 과 양방향 모델 비교 학습

Train - LSTM

Unit에 따른 LSTM 비교 by FastText

	epoch	accuracy	val_accuracy	loss	val_loss
unit = 30	18	0.8156	0.5951	0.4538	1.3361
unit = 50	18	0.8158	0.5887	0.4627	1.3417
unit = 128	16	0.8394	0.6169	0.3862	1.2756

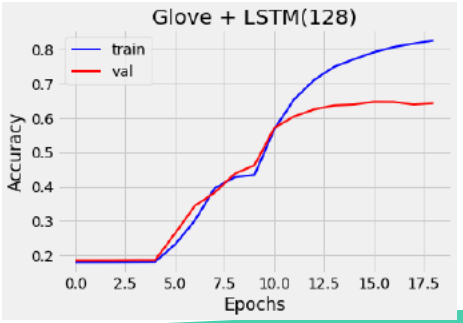


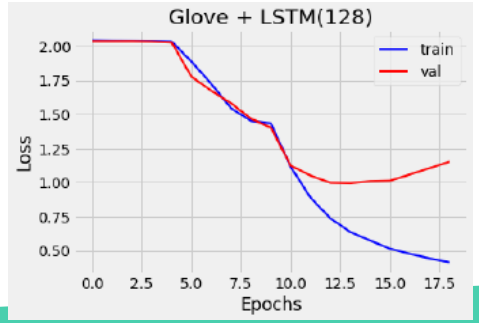


Train - LSTM

Unit에 따른 LSTM 비교 by Glove

	epoch	accuracy	val_accuracy	loss	val_loss
unit = 30	19	0.7542	0.5590	0.6682	1.3197
unit = 50	17	0.7740	0.5521	0.6083	1.3739
unit = 128	19	0.8260	0.6430	0.4156	1.1537

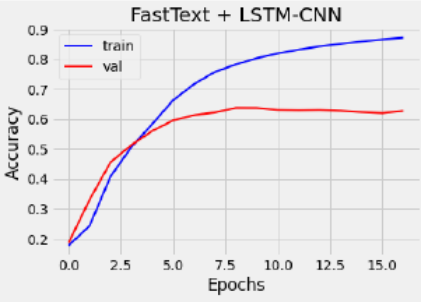


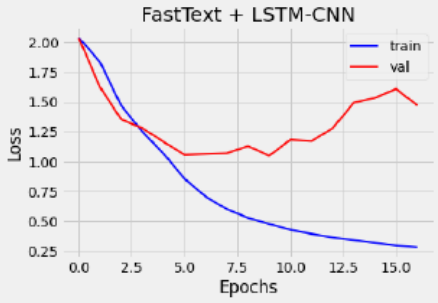


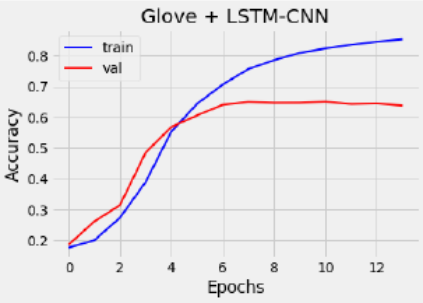
Train - LSTM

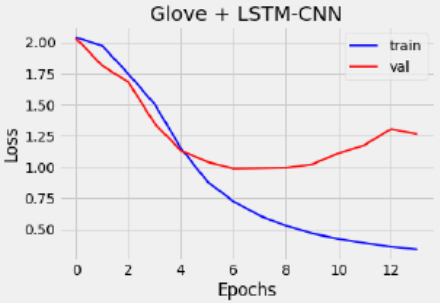
LSTM – CNN 비교

	epoch	accuracy	val_accuracy	loss	val_loss
FastText	17	0.8714	0.6268	0.2814	1.4708
Glove	14	0.8531	0.6377	0.3390	1.2650

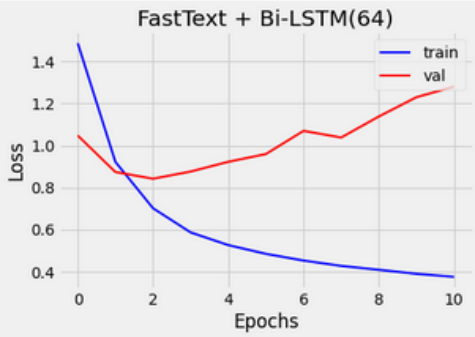


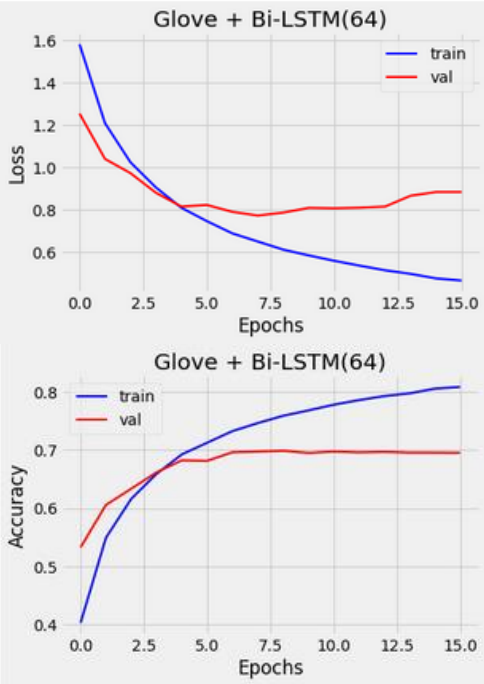






Train – Bi-LSTM





	Val_loss	Val_acc
FastText	0.8723	0.6695
Glove	0.7960	0.6866

Train – Bi-LSTM + Attention

Attention Mechanism

- RNN의 대표적인 단점인 장기의존성 문제를 해결하기 위한 기법

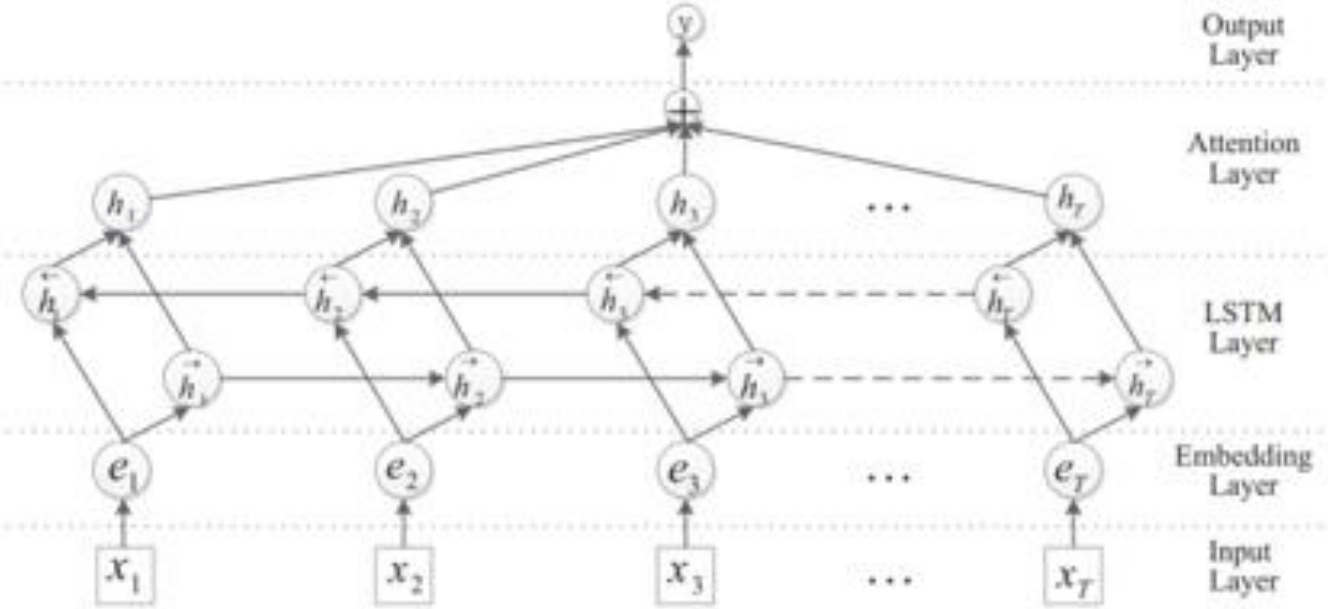
- 핵심 아이디어

  매 스텝마다 입력을 참고해, 연관 부분을 집중(attention)하여 결과를 예측하는데 기여

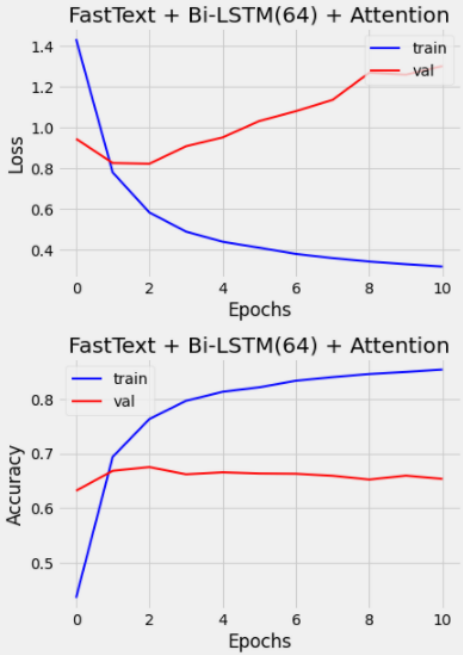
- 과정

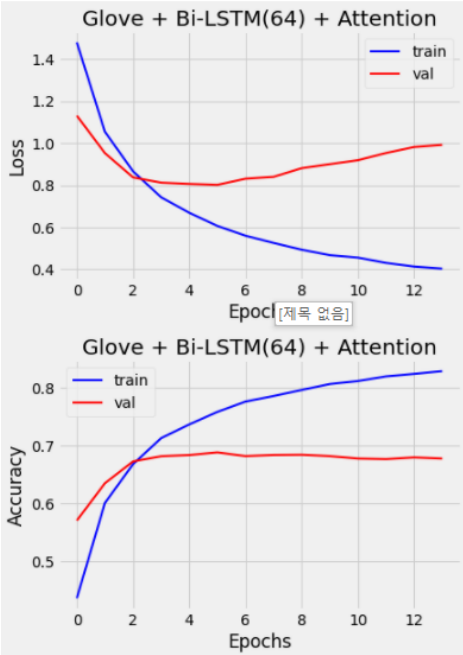
  1) Attention Score : 매 스텝 디코더의 은닉 계층과 인코더의 은닉 계층의 유사도 계산
  2) Attention Distribution : Attention Score 를 활용해 소프트 맥스 함수를 취해 계산
  3) Attention Value : Attention Distribution 과 인코더의 Context vector를 곱해 계산
  4) Attention Value와 디코더의 출력 벡터를 연결(Concatenate)하여 새로운 벡터 생성
  이를 출력 층의 입력으로 사용

Train – Bi-LSTM + Attention



Train – Bi-LSTM + Attention

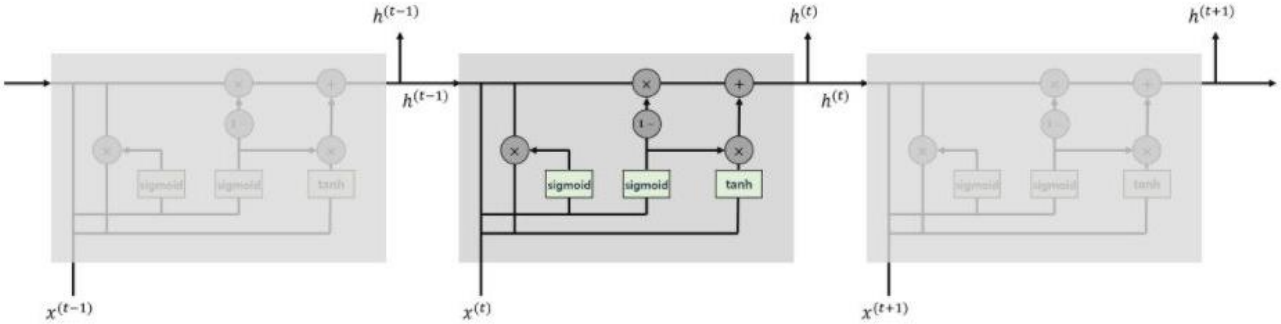




	Val_loss	Val_acc
FastText	0.7913	0.6842
Glove	0.7893	0.6959

Train – GRU

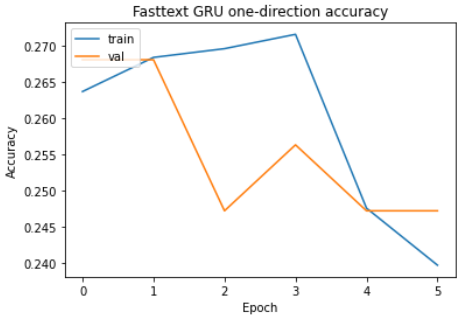
GRU : Reset gate, Update gate, Candidate, Hidden layer 로 구성

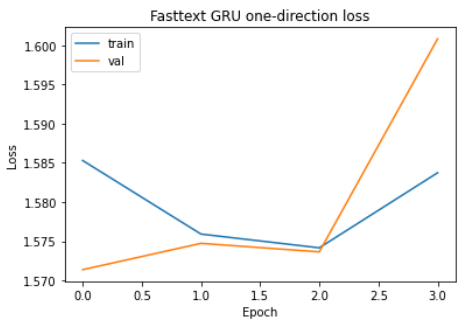


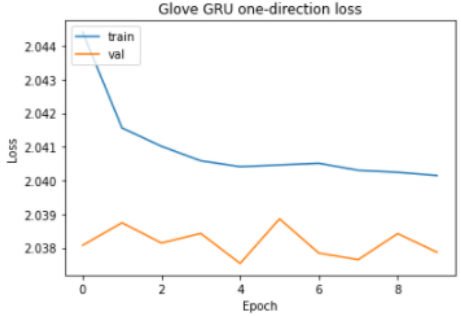
$$r^{(t)} = \sigma\left(W_r h^{(t-1)} + U_r x^{(t)}\right)$$
$$u^{(t)} = \sigma\left(W_u h^{(t-1)} + U_u x^{(t)}\right)$$
$$\tilde{h}^{(t)} = \tau\left(W h^{(t-1)} * r^{(t)} + U x^{(t)}\right)$$
$$h^{(t)} = (1 - u^{(t)}) * h^{(t-1)} + u^{(t)} * \tilde{h}^{(t)}$$

Train – GRU

단방향 GRU 학습 결과



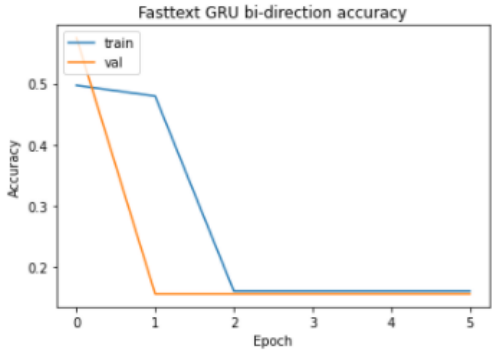


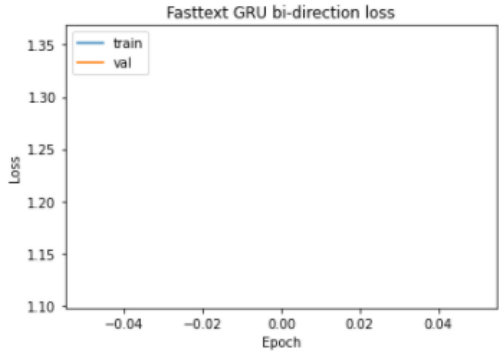


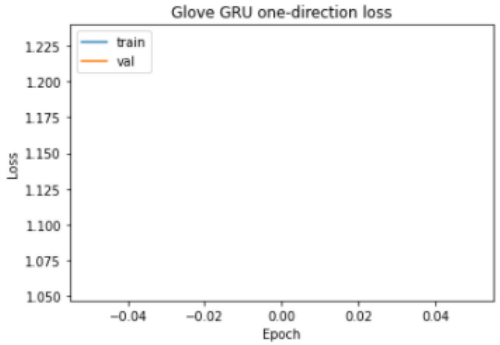
	Val_loss	Val_acc
FastText	2.0380	0.1849
Glove	2.0379	0.1849

Train – GRU

양방향 GRU 학습 결과







	Val_loss	Val_acc
FastText	Nan	0.1554
Glove	Nan	0.1554

Conclusion

- CNN 학습 결과

  ✓ 데이터 셋에 따른 학습 결과 : Original Dataset 〉 Custom Dataset (평균 10 %)

  1) 대회를 목표로 만들어진 Original Dataset에 비해 추가한 데이터들에 대한 전처리 미흡 가능성
  2) Odin 과 같은 특수한 명사에 의해 기존 보다 정확도가 떨어졌을 가능성

  ✓ FastText's best match : 기본 CNN / Glove's best match : Long CNN

  임베딩 모델에 따라서 CNN 모델의 구성이 영향을 주며, 각 임베딩 모델의 최적화 CNN 설계 필요

Conclusion

- LSTM 학습 결과

  Test data에 대해서 LSTM-CNN 모델의 성능이 좋았지만, Validation data에 대한 결과와 차이가 나기 때문에, 이에 대한 파라미터 조정, 층, 다른 모델과의 결합을 통한 연구 필요

- Bi - LSTM 학습 결과

  모든 임베딩 모델에 대해, LSTM 보다 높은 성능을 확인
  또한 Bi – LSTM + Attention Mechanism에 대해서 가장 높은 성능을 확인
  더욱 높은 학습 결과를 위해서는 다양한 파라미터 조정을 통한 연구 필요

Conclusion

- GRU 학습 결과
  - ✓ 학습 초기에 활성화 함수 'relu' 사용 시 1 epoch 당 약 20분 소요

    이후 GRU 의 Default 활성화 함수인 'tanh' 사용 (1 epoch 당 3분 소요)

  - ✓ 정확도 개선 실패 이유

    1) 적절한 Learning rate, GRU parameter 에 대한 이해 부족

    2) 모델을 구성하는 계층에 대한 배치 실패 혹은 너무 적은 계층

Q & A