Rush 1
Grammar Error Correction
한국어 문법 교정

AI RUSH
TEAM yoncat
최연웅, 임도연

Introduction

- 한국어 문법 교정

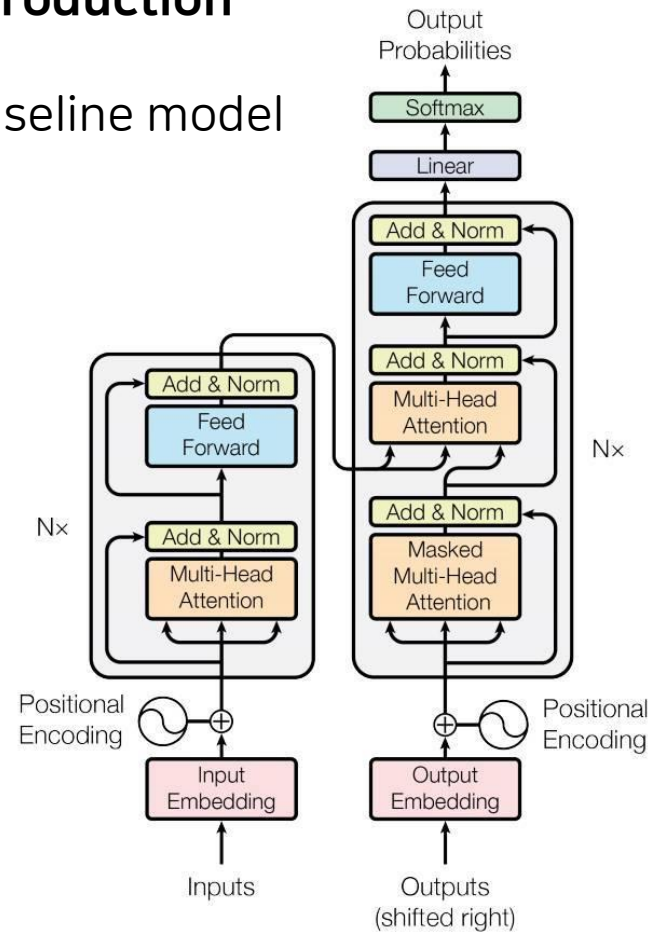
외않됀데? ⟶ 왜 안 된대?

- 에러 유형

Punctuation	이게 최고다	이게 최고다.
Spacing	내마음 바람에실려.	내 마음 바람에 실려.
Recommendation	요오런 뽀짝 트리가 있었다 ~	이런 귀여운 트리가 있었다.
Typos	물론 내 맴대루	물론 내 마음대로.
Honorific	너무 잘하셨어요~!	너무 잘하셨습니다!
Tense	올해 너무 덥다고 해서 미리 에어컨 설치해 놨는데	올해 무척 더울 거라고 해서 미리 에어컨 설치 해놨는데.

Introduction

- Baseline model



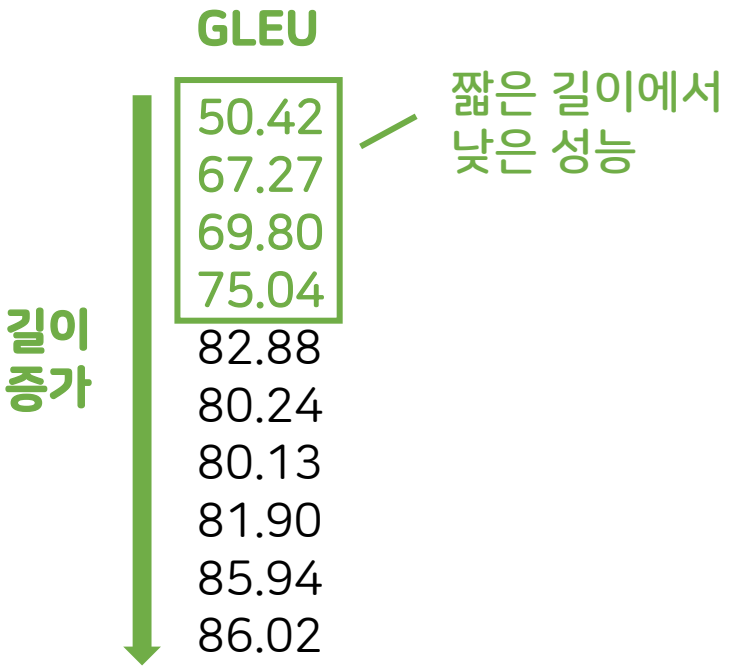
Transformer seq2seq model

- Dataset
  - 출처
    - Naver Blog
  - 크기
    - Labeled: 19K
    - Unlabeled: 20K

Analysis & Methodology

- 길이 별 성능

  Validation 1000개 길이 순 정렬,
  100개씩 평가



- 원인

  1. 완벽한 문장이 아닌 경우
  2. 너무 짧은 경우

  문장의 끝을 인식하지 못해,
  단어 반복 문제가 발생

  예)
  소름 돋는 부분
  ⇒ 소름 돋는 부분 부분 부분 부분.

Analysis & Methodology

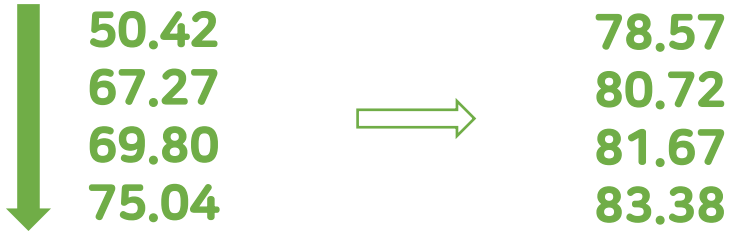
- 길이 별 성능 차이 해결 방법

1. Input 끝에 <EOS>를 주어, 문장의 끝을 인식시킴
   - 태스크 특성상, input과 output의 문장의 끝이 유사하다는 점에서 적용가능

2. Output lengt를 input length의 1.15배로 강제함
   - 문법 교정에서 문장의 길이는 크게 바뀌지 않는다는 점을 이용

3. 예측 시 <EOS> 토큰의 prediction probability를 1.2배 높여줌

길이 순 정렬 batch
(각 100개 instance)



Analysis & Methodology

- 에러 별 개수 및 성능

에러유형	Punctuation	spacing	Recomm.	typos	honorific	tense
에러 갯수	11676	11676	7547	1533	1533	160
GLEU	84.65	85.32	78.31	80.41	78.74	77.44

에러 개수가 많음 에러 성능이 낮음

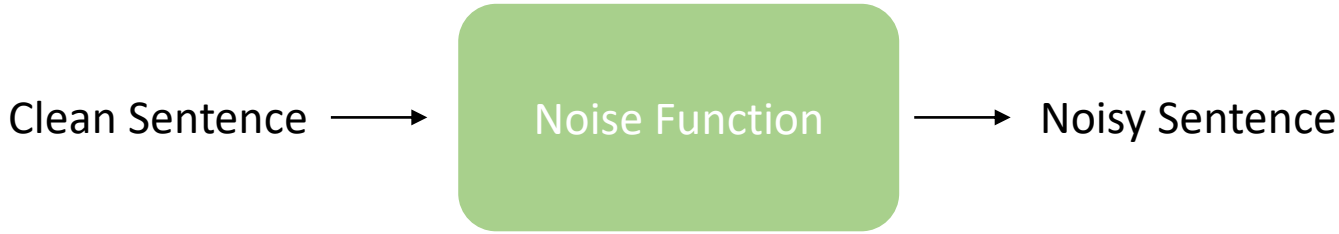
- 원인

학습 가능한 데이터 셋이 19K로,
특히 recommendation/ typo 에러유형을 다양하게 볼 기회가 없음*

* 직관적으로도 그렇고 유명 post-editing challenge에 참여한 다수의 팀들이 논문에서 언급한 바가 있음

Analysis & Methodology

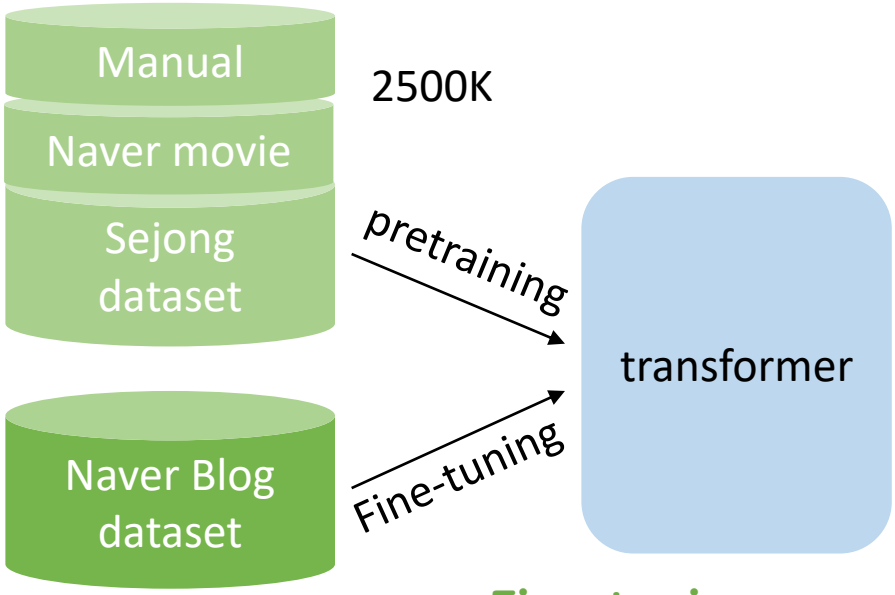
- 특정 에러 성능 향상 방법



- Dataset
  - 2000년 대 초 국립국어원에서 배포한 세종 corpus
    - 생활에 밀접하고 다양한 주제로 네이버 블로그와 유사하다고 여겨짐
    - 2000년대 초반에 발간된 잡지 위주로, 옛말과 문어체가 다수임
  - 2000K의 크기
  - Hanspell 라이브러리를 통해 클렌징 작업을 함

Analysis & Methodology

- 특정 에러 성능 향상 방법



Fine-tuning
- Dropout 비율 줄임
- Learning rate 줄임

최종 모델 선정

- Labeled data의 일부분을 validation set으로 사용해 정성 및 정량평가
- Unlabeled data의 prediction 결과를 통한 정성평가

Analysis & Methodology

- 특정 에러 성능 향상 방법
- Noise Function
  - 한글 특성에 맞도록, punctuation/spacing/typo/recommendation 위주의 noise를 생성함

1. **Puntuation**: . ! ? 를 랜덤하게 문장 끝에 랜덤한 개수로 붙임
2. **Spacing**: 임의의 위치에 space를 삽입 혹은 이미 있는 space를 삭제
3. **Typo**:
   - 자모를 해체하여 키보드 기준으로 가까운 자모로 대체
   - 발음이 나는 대로 문장을 바꿈 (g2pk 라이브러리 이용)
   - 문장 끝 문자에 ㅁ,ㄴ,ㅇ,ㄹ,ㅅ,ㅋ,ㅎ 받침을 임의로 삽입 예) 했다 → 했당
   - 연음의 경우 받침을 다음 ㅇ으로 옮김
4. **Recommendation**
   - 자주 헷갈리는 단어들 위주로 dictionary를 만들어 치환
   - 비속어를 일정한 확률로 삽입
   - 무의미한 알파벳 배열을 일정한 확률로 삽입

Analysis & Methodology

- 특정 에러 성능 향상 방법
- Noise Function
  - 한글 특성에 맞도록, punctuation/spacing/typo/recommendation 위주의 noise를 생성함

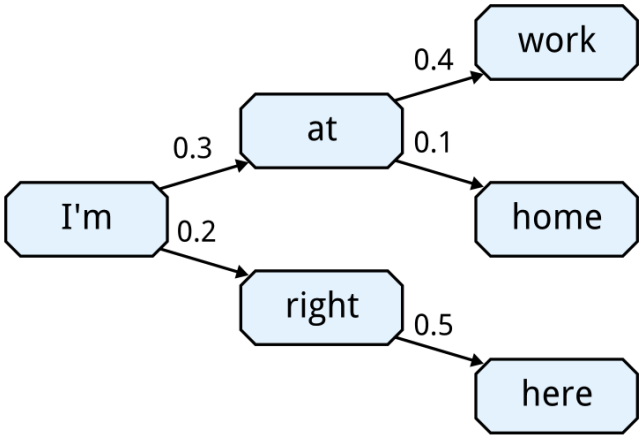
5. 기타 전통적 noise generation

- 국소적 글자재배치
- 임의의 문자 삽입, 삭제, 치환

에러유형	Punctuation	spacing	Recomm.	typos	honorific	tense
Baseline	84.65	85.32	78.31	80.41	**78.74**	77.44
Pretrained	**87.05**	**88.61**	**80.69**	**85.32**	78.01	**82.57**

Analysis & Methodology

- 기타 성능 향상 기법
- Beam Search



- Decoder에서 token하나를 prediction할 때마다, 확률이 높은 상위 5개의 candidate들을 유지하여 최종적으로 가장 높은 확률의 sequence를 반환

성능 향상: 약 0.2 ~ 0.3의 GLEU score향상

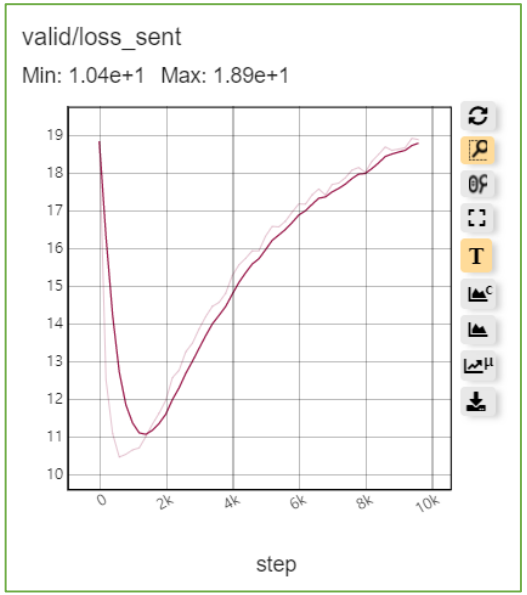
서비스 속도: 3주차 test set 기준 6.49배 차이 (47.68초 / 309.51초)

Analysis & Methodology

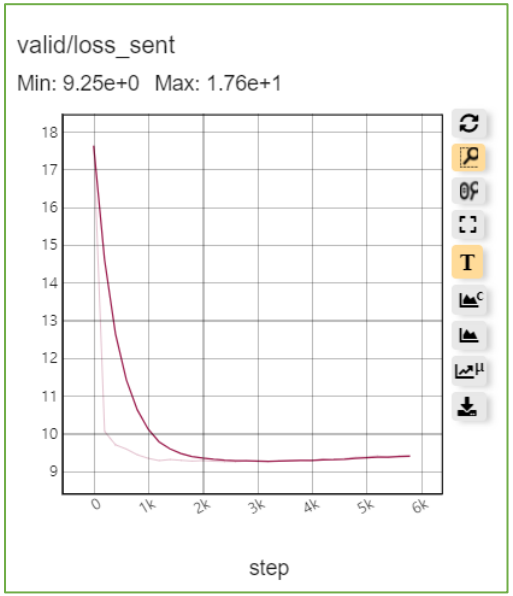
- 기타 성능 향상 기법
- Vocabulary 크기 조정
  - 최대한 많은 clean token과 빈도 수 높은 noise token을 커버하되, vocab이 너무 커서 sparse해지지 않도록 조정
  - Sejong clean 및 noisy data 의 최빈 2000음절 토큰
    + Naver blog labeled (clean 및 noisy)와 unlabeled의 99.9% 음절을 커버하는 vocabulary 선정
  - 총 2113개의 token
- <UNK> 토큰 그대로 copy해서 대응

Analysis & Methodology

- 기타 성능 향상 기법
- Hyperparameter tuning
  - Learning rate – 0.0001 (fine-tuning)
  - Dropout – 0.15
  - Batch size - 256

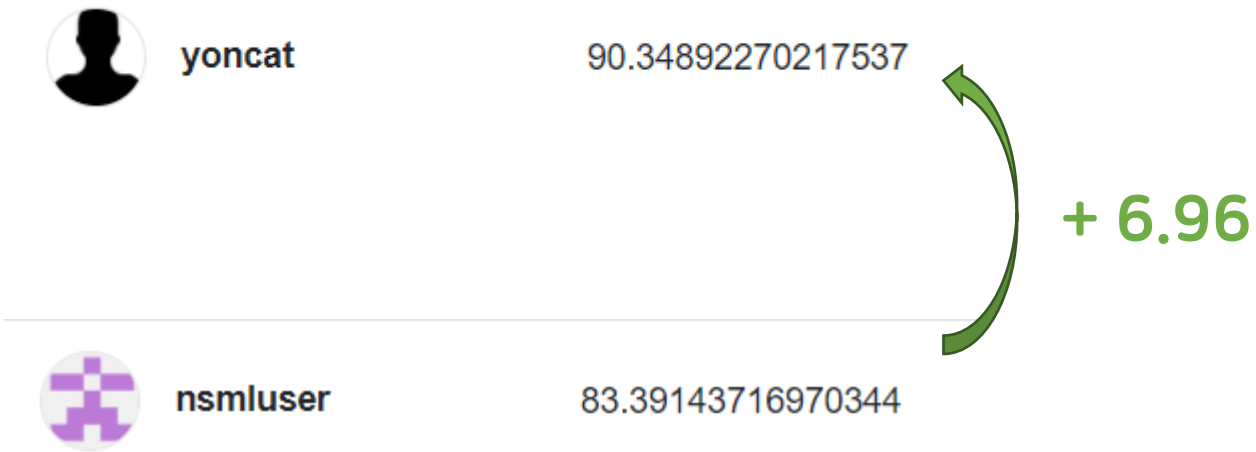


lr 조절



Analysis & Methodology

- Baseline 대비 최종 성능 향상



Trials & Errors

- KoBert, KcBert, KoELECTRA 모델 적용
  ➔ 19K데이터만 있을 때 적용. Baseline보다 떨어짐. 데이터 부족에 비해 모델이 너무 복잡했던 것이 문제였던 것으로 예상
- KoBert Tokenizer적용
  ➔ pretrained 되어있는 8002 SentencePiece tokenizer 사용, 음절단위보다 의미있는 문자덩어리가 교정에 더 유리할 것이라 생각해서 적용했으나 성능이 떨어짐. 우리 corpus와 맞지 않는 등 여러가지 가능성이 있으나 fine tuning해보진 않았음
- 길이 조정 테크닉
  - 한국어 문장에서 space가 전체 길이의 20%정도 되는 것에 착안하여 극도로 띄어쓰기가 안되어 있는 경우, (길이의 20%가 안되는 경우) output length 를 더 허용해줌
  - Beamsearch에서 length가 길 수록 불리한데, depenalize를 해줌
- Labeled data oversampling
  - 성능이 3~4%씩 떨어짐, clean 문장에 과적합 되는 것으로 판단

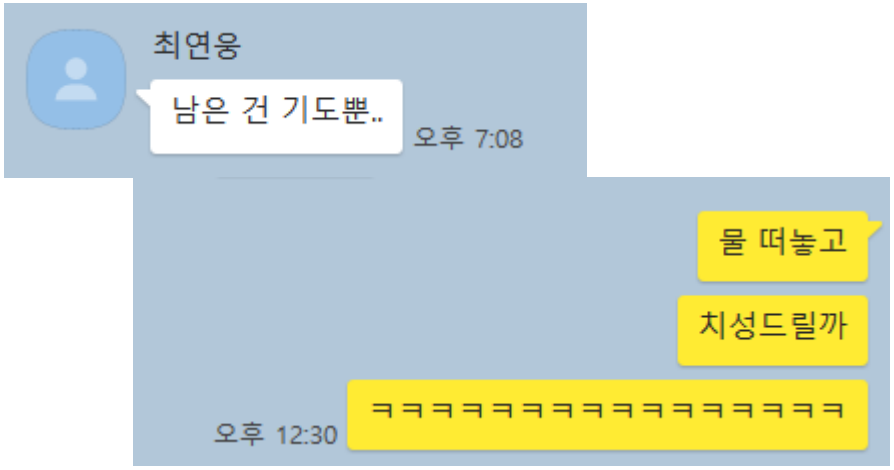
Trials & Errors

- Unlabeled data를 이용한 Masked LM pretraining
- Input sentence의 이모티콘 등을 지우는 preprocessing
- 성능이 높은 checkpoint들의 model weight averaging으로 generalization
- Backtranslation 으로 noise 생성

...

그리고… 기도

```
nsml submit t0005/rush1-3/507
```



(둘 다 무교입니다…)

Q&A

APPENDIX

1등 하고 싶어서 했던 하드코딩

Confusing Dictionary

```
'아': ['애'],
'돼': ['되','대'],
'되': ['돼'],
'된': ['됀','댄'],
'안': ['않'],
'않': ['안'],
'왜': ['외'],
'원': ['one'],
'가격': ['price'],
'사이즈': ['싸이즈'],
'주문': ['오더', 'order'],
'대박': ['대애박', '대애애박', '대애애애박'],
'피시': ['피', '피씨', 'PC'],
'바람': ['바램'],
'바라': ['바래'],
```

- 자주 나오는 오류를 noise로 변환.
- 모델이 오류를 clean text로 변환하길 기도하면서 pretrain.
- 그리고 실제로 된다.

[주 추천메뉴는 산채비빔흰밥 one 1인분에 1만one], [주 추천 메뉴는 산채비빔흰밥 원 1인분에 1만 원.], [주 추천 메뉴는 산채비빔흰밥으로 1인분에 1만 원.]

noisy | prediction | clean

무수한 시도들…

Rank	Name	Score	Model	Args
1	 yoncat	90.34892270217537	t0005/rush1-3/507/best	--resubmit 502 --beam_width 5 --seed 2020 --min_margin 2 --eos_multiple 1.2
2	 yoncat	90.34749703155009	t0005/rush1-3/514/best	
3	 yoncat	90.34271972567761	t0005/rush1-3/652/best	--resubmit 507 --beamsearch --beam_width 5 --seed 2020 --min_margin 2 --eos_multiple 1.1
4	 yoncat	90.24411101767559	t0005/rush1-3/551/best	--resubmit 499 --beam_width 5 --seed 2020 --min_margin 2 --eos_multiple 1.2
5	 yoncat	90.21446875375274	t0005/rush1-3/631/best	--resubmit 507 --beam_width 5 --seed 2020 --min_margin 2 --eos_multiple 1.2 --beam_length_penalty 0.5
6	 yoncat	90.18039700769236	t0005/rush1-3/638/best	--resubmit 507 --beam_width 5 --seed 2020 --min_margin 2 --eos_multiple 1.1
7	 yoncat	90.17648268462169	t0005/rush1-3/445/best	--load_model pretrained_404.pt --beam_width 5 --num_warmup_steps 10 --seed 2020
8	 yoncat	90.14087230115169	t0005/rush1-3/450/best	--beam_width 5 --num_warmup_steps 10 --eval_interval 500 --seed 2020 --load_model pretrained_404.pt
9	 yoncat	90.1370197442705	t0005/rush1-3/441/best	--load_model pretrained_404.pt --beam_width 5 --num_warmup_steps 10
10	 yoncat	90.1021476414567	t0005/rush1-3/590/best	--resubmit 507 --beam_width 5 --seed 2020 --min_margin 0 --eos_multiple 1.0 --beam_length_penalty 0.5