한국어 문장의
의미적 텍스트 유사도 분석

: KLUE-STS Benchmark Dataset을 이용하여

장 수 림 think.percento@gmail.com
담당 / 전처리, 데이터 증강, SBERT모델링

분석 정의

음식점을 돌아다니며 고객 서비스를 제공하는 **서빙 로봇**에게 다음의 질문을 하려합니다.

결제는 어디에서 하나요? ≒ **계산하는 곳이 어디인지 알려주세요.**

과연 로봇은 두 질문에 같은 답변을 준비할 수 있을까요?

STS(Semantic Textual Similarity) ; **텍스트의 의미적 유사도**를 평가하는 NLU Task

예시와 같이 통사적으로 다르지만, 의미적으로는 같은 문장을
기계가 더 잘 이해한다면 **서비스의 효율**이 제고될 것

- 모델이 두 문장의 **의미적 친밀도**를 얼마나 잘 잡아내는가?
- 모델이 문장의 **의미적 표현**을 얼마나 잘 구현하는가?

⇒ **Metric**: f1 score, pearson' r

KLUE-STS 개요

KLUE-STS은 리뷰, 기사, 스마트홈 디바이스의 발화 코퍼스를 가공한 한국어 의미 유사도 데이터셋.
문장 쌍은 꼬꼬마 기준 약 14개의 토큰으로 구성되며, 이는 KorSTS(kakao brain)에 비해 두 배 정도 긴 분량이다.
의미적 유사도를 나타내는 **레이블**은 복수의 작업자에 의해 측정되며, 0 (상이함) ←--→ 5 (동등함)의 척도를 갖는다.

KLUE-STS	Train	Dev.
Source	Airbnb (Review), Policy (News), ParaKQC (Smart home Query)	
# Examples	11,668	519
Avg. # tokens	14.5	14.1

[표3] Summary of KLUE-STS Datasets

Feature	Contents	Data type
guid	'klue-sts-v1_train_00000'	str
sentence1	'숙소 위치는 찾기 쉽고 일반적인 한국의 반지하 숙소입니다.'	str
sentence2	'숙박시설의 위치는 쉽게 찾을 수 있고 한국의 대표적인 반지하 숙박 시설입니다.'	str
labels	{'binary-label': 1, 'label': 3.7, 'real-label': 3.714285714285714}	Int / float64
source	'airbnb-rtt'	str

[표4] Sample of KLUE-STS Datasets

KLUE-STS 탐색

1 텍스트 정제

KLUE-STS는 이미 Copora 정제를 아래와 같이 수행

⇒ 별도의 텍스트 전처리 과정은 생략하자

- Noise Filtering

  한국어가 아닌 텍스트, 해시태그(#), HTML태그, 빈 괄호(), 연속 공백, 기자 및 언론 정보, 이미지, 출처에 대한 정보, 저작권 태그 제거

- Toxic Content Removal

  Korean hate speech 데이터셋을 활용한 성별 편향과 혐오 발언 탐지 훈련으로 유해 콘텐츠 제거

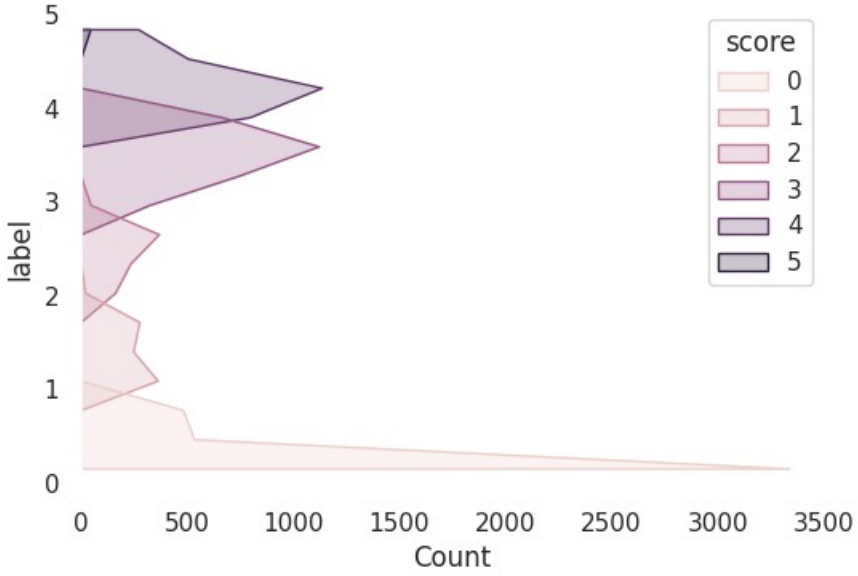
- PII Removal

  개인정보가 포함된 텍스트를 제거. 공인으로 간주되지 않는 개인정보 (이름, 주민등록번호, 전화번호, 은행 계좌 등), 이메일 주소, URL, @name 등의 텍스트를 정규표현식으로 제거

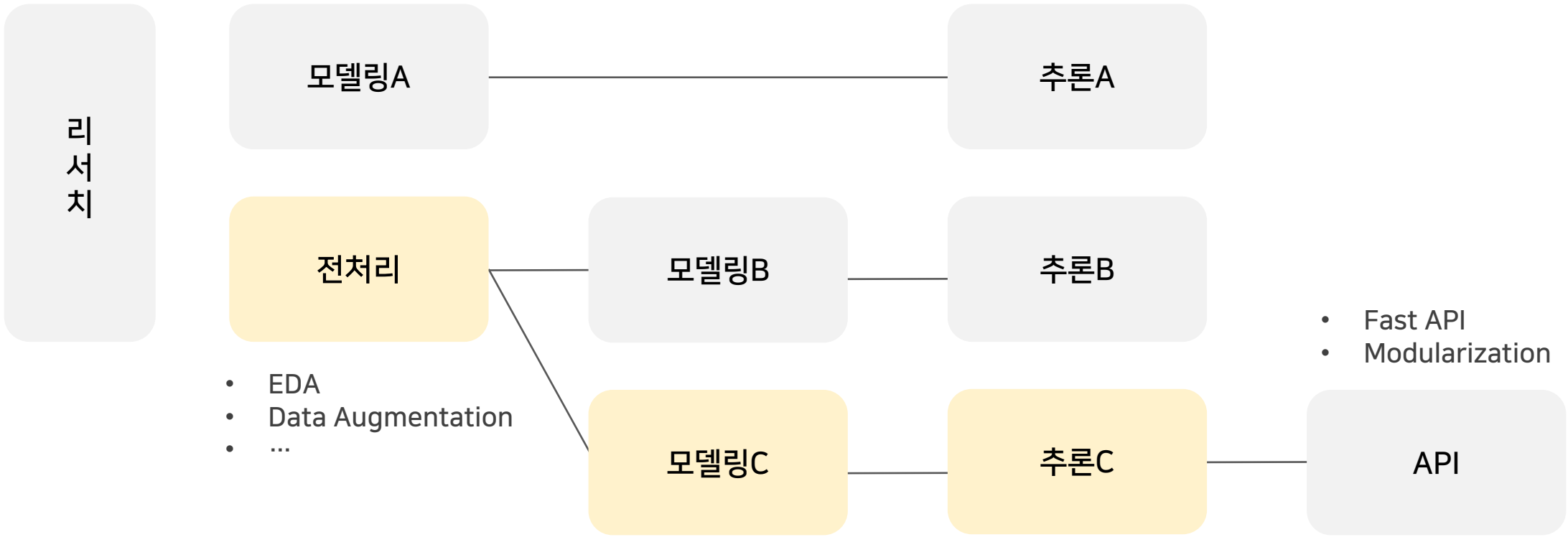
2 레이블 불균형

0~1점 대는 지나치고 많고, 1~3점 대는 적다

⇒ 데이터 증강 기법을 이용해 불균형 개선하자



분석 로드맵

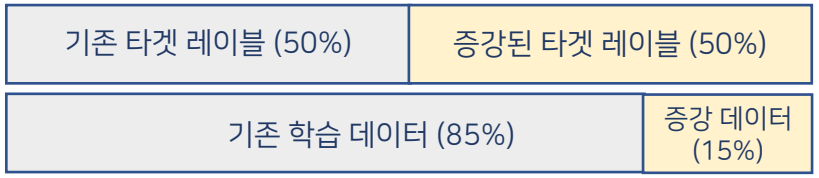

리서치
모델링A
추론A
전처리
모델링B
추론B
• EDA
• Data Augmentation
• ...
모델링C
추론C
API
• Fast API
• Modularization

Data Augmentation

데이터 증강 기법의 합리적 선정을 위해, **증강 과정에서 의미 유사성을 지킬 수 있는 방법인지**를 중점적으로 고려하였다.

- EDA(Easy Data Augmentation) Random Swap

  문장 중 임의의 두 단어 위치를 바꾸어 문장 생성



  [가설] 한국어는 영어에 비해 어순에 의미 의존이 적으므로, 데이터 증강 시 모델의 예측 성능을 높일 것이다.

  [적용] ko-EDA (alpha=0.1, num_aug=4) ⇒ 비교군에 비해 f1 score -0.014 하락

- Back Translation

  타겟 언어를 타국어로 번역한 후에 다시 타겟 언어로 번역한 문장 생성

  [가설] 역번역 과정에서 토큰의 의미가 보존되어, 데이터 증강 시 모델의 예측 성능을 높일 것이다.

  [적용] pororo (ko > en > ko) ⇒ 비교군에 비해 f1 score -0.16 하락

- **왜 데이터증강이 효과적이지 않았는가?**

Confusion matrix 살펴보니, 오히려 타겟 레이블이 예측을 잘 수행했음. 모델이 무얼 잘 예측할지 파악해 전략적으로 증강 기법 적용해야.

Data Preprocessing

학습 속도 향상(speed [it/s], time spent [time/epoch]) 및 **예측 성능 향상**(pearson'r, f1)을 위한 전처리 고려사항은 다음과 같다.

- Tokenizer
  - Dynamic padding

    배치 별로 문장 최대 길이를 맞춰, Fixed padding 에 비해 약 30% 학습 속도 향상

- Text preprocessing

  앞선 언급과 같이, 학습 데이터가 이미 충분히 정제되었다 판단하여 별도의 추가적인 텍스트 정제는 수행하지 않음.
  다음의 전처리 기법들을 고려했으나 데이터에 맞지 않아 반려함. 추후 다른 STS task 다룰 때 적용해볼 것을 제안.

  - py-hanspell : 문장 내 맞춤법을 교정 ⇒ '기계가 generic한 표현형만 학습하는 것이 바람직한가?' (X)
  - soynlp : 자체 normalizer 이용해 의미 없이 중복되는 문자 및 이모티콘 등 제거

    ⇒ review 데이터에 노이즈 텍스트가 있을 것으로 예상했지만 이미 처리되어 사용하지 않음
  - soyspacing : 띄어쓰기 오류를 학습해 교정. PyKoSpacing 에 비해 가볍고 빠름

    ⇒ 띄어쓰기 오류와 '언어 패턴'은 다름. 데이터의 생성자가 상이한 조건에서는 이를 구별하기 어려움

모델링 요약

데이터 마다 고유한 내재적 특성이 있을 것이다. 따라서, 그 **특성을 더 효과적으로 파악하는 모델**을 찾기 위해
단일 모델의 탐색보다 다음의 **4가지 모델 방법론** 각각을 적용해가며 최적의 모델을 선정하는 전략을 사용

BERT	Dev f1
BERT baseline (plm=klue-bert-base, opt=AdamW, batch_size=32, lr=2e-5, epoch=50)	0.510
BERT + pooler concat	0.422
BERT + abs(u,v)	0.477
BERT + pooler element wise product	0.489
BERT + siamese-network like	0.522
BERT + scaled cosine similarity	**0.662**

[표5] Experiments Summary of BERT

RoBERTa	Dev f1
RoBERTa +FCN (plm=klue-robert-base)	**0.590**

[표6] Experiments Summary of RoBERTa

Sim-CSE	Dev f1
Sim-CSE + Back Translation (using kor-nli + klue-nli)	**0.883**

[표7] Experiments Summary of Sim-CSE

⇒ 과제 규정에 따라, 외부 데이터를 학습한 본 모델은 제출하지 않고 인사이트를 위해 남겨두기로 결정

SBERT	Dev f1
SRoBERTa baseline (plm=klue-robert-base, opt=AdamW, batch_size=16, lr=2e-5, epoch=4)	0.834
SRoBERTa + Back Translation	0.67
SRoBERTa + Random Swap	0.82
SRoBERTa + batch_size=8	**0.841**

[표8] Experiments Summary of Sim-CSE

Sentence-Bert

기존 BERT가 지닌 한계를 분명하게 개선하고, 타 문장 임베딩 기법보다 학습시간이 빠르며* 단순한 코사인 유사도를 이용**

*random weight initialization 대신, pre-trained BERT 사용 / **유사도 점수에 Regression function을 이용한 기존 모델은 부하가 커지면 unscalable

Embedding and Training

1️⃣ BERT의 입력으로 (문장 A, 문장 B)를 넣고, **평균** 또는 맥스 풀링*을 통해서 각각에 대한 문장 임베딩 벡터를 얻는다 .

* **평균 풀링** [채택] : 모든 단어의 전반적인 의미를 반영 vs **맥스 풀링**: 중요한 단어의 의미를 반영

2️⃣ 생성된 두 벡터 (u, v) 의 코사인 유사도를 구한다.

3️⃣ 해당 벡터 유사도와 레이블 유사도와의 평균 제곱 오차(Mean Squared Error, MSE)를 최소화하는 방식으로 학습한다.

Objective Functions = {문장 쌍 분류 $o=softmax(W_t(u,v,|u-v|))$, 문장 쌍 회귀 $o=\sigma(cosine\_similarity(u,v))$}

두 Task의 목적함수를 모두 학습하거나 하나만 학습할 수 있다.

연구 결과에 의하면, NLI Task 학습한 뒤 STS Task 학습한 SBERT의 성능이 가장 좋았으나 규정에 따라 **STS만 학습**하였다.

모델 성과

분석 대상인 KLUE-STS Datasets 토대로 총 14회의 모델링 실험을 진행
그중 가장 높은 성과를 낸 모델은 **SentenceBert**(이하 SBERT) 기반의 학습 모델로,
Dev set 기준 성능은 **Pearson' r** - 88.9 / **F1 score** - 84.1

모델이 예측한 문장 유사도는 작업자가 측정한 **실제 유사도와 강한 상관관계**를 지니며, 모델은 **높은 정밀도 및 재현율**을 보였다.

No.	Language Model	Pearson' r	F1
1	SRoBERTa	**88.9**	**84.1**
2	Sim-CSE	88.3	81.0
3	RoBERTa	70.0	59.0
4	BERT	33.0	66.2

[표1] 언어 모델 별 분석 결과표
(각 모델의 실험 결과 중 가장 높은 f1 score 기록을 기준으로 함)

No.	Language Model	Data Augmentation	Adoption or not
1	SRoBERTa	Random Swap	X
2	SRoBERTa	Back Translation	X
3	Sim-CSE	Back Translation	O

[표2] 분석에 이용한 데이터 증강 기법의 채택 여부

더 나아가야할 지점들

- **더 가볍게, 더 빠르게 설계하기**
  - 정확도가 1% 낮더라도 0.1초 빨리 서빙할 수 있는 선택지를 알아야한다

  Float point 32 bit >16 bit 변환처럼 설계상의 작은 디테일로 연산이 수월해지도록 해야한다.

- **Augmentation 시야 넓히기**
  - 데이터셋뿐만 아니라, 모델 아키텍쳐의 Augmentation을 고려해야한다

  예를 들어, Sim-CSE는 Contrative Learning 수행할 때 서로 다른 random seed를 사용하여 dropout 레이어의 랜덤성을 변경해 positive sample을 생성한다.

- **모델의 지표를 평가가 아닌 추론에도 활용하기**
  - 단순히 높은 숫자를 얻으려는 것이 아닌, 데이터의 고유한 특성을 추론하려는 태도의 전환

  도메인 데이터에 대한 분석을 진행할 때, 정량적/정성적 지표를 통해 해당 방법론을 적용하는 것에 이점이 있는지를 추론하고 개선점을 창출하는 접근이 필요하다. 사전학습된 모델 사용은 언제나 바람직한가? 많은 양의 데이터가 지표를 보정해주기도 하지만, Anisotropic에 빠져 과적합 문제를 피할 수 없을지 모른다.

출처

최종 채택된 모델에 사용된 데이터 및 모델에 관한 출처만 표기함

- Sungjoon Park, et al., (2021) KLUE: Korean Language Understanding Evaluation [Link]
- Reimers , et al., (2019) Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, Association for Computational Linguistics [Link]