

허깅페이스야 니답시 살어야

한국어 표준어-방언 번역기와 방언을 구사하는 챗봇 구현

Team | NLP 4팀

18기 강동현 19기 정종락 20기 김채원

CONTENTS

01

프로젝트 목표

- 프로젝트 목표
- 프로젝트 방향성

02

데이터 수집 및 전처리

- 방언 데이터
- 대화 데이터

03

Model

- 표준어 질문 - 사투리 대답
- 표준어 질문 - 표준어 대답
- 표준어 대답 사투리 번역
- 학습 모델 결합

04

Result & Discussion

- 프로젝트 의의
- 한계와 개선 방향



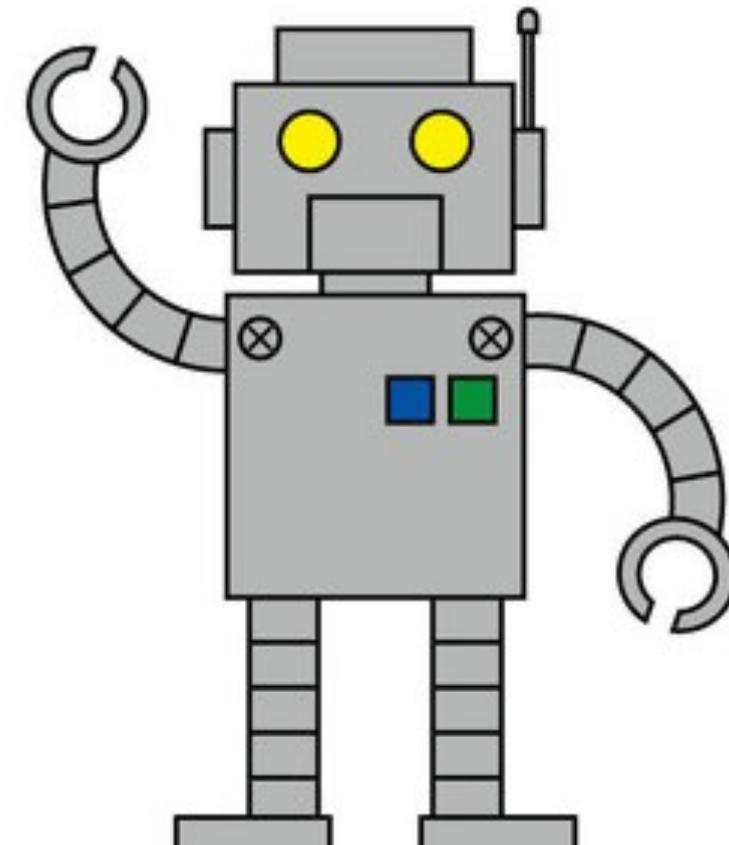
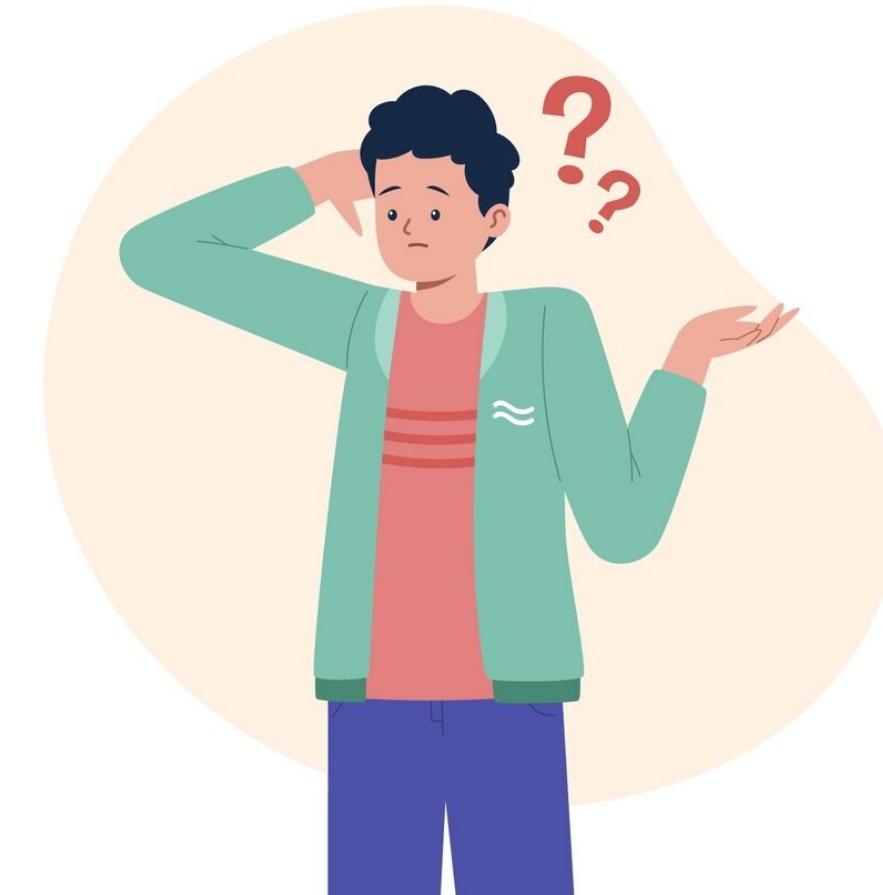
01. 프로젝트 목표

01. 프로젝트 목표

“ 지역 방언을 사용해 질문에 응답하는 챗봇 개발 ”

오늘 저녁 메뉴는 뭐가 좋을까?

흠, 이왕 허는 김에 갈치조림에 몸국 허곡, 아니면
고사리육개장 어떻햄수과? 허민 든든하고 …



01. 프로젝트 방향성

필요한 데이터셋

대화 데이터

지역별 방언 - 표준어상 데이터

...

파이프라인 설계

Task 1

Task 2

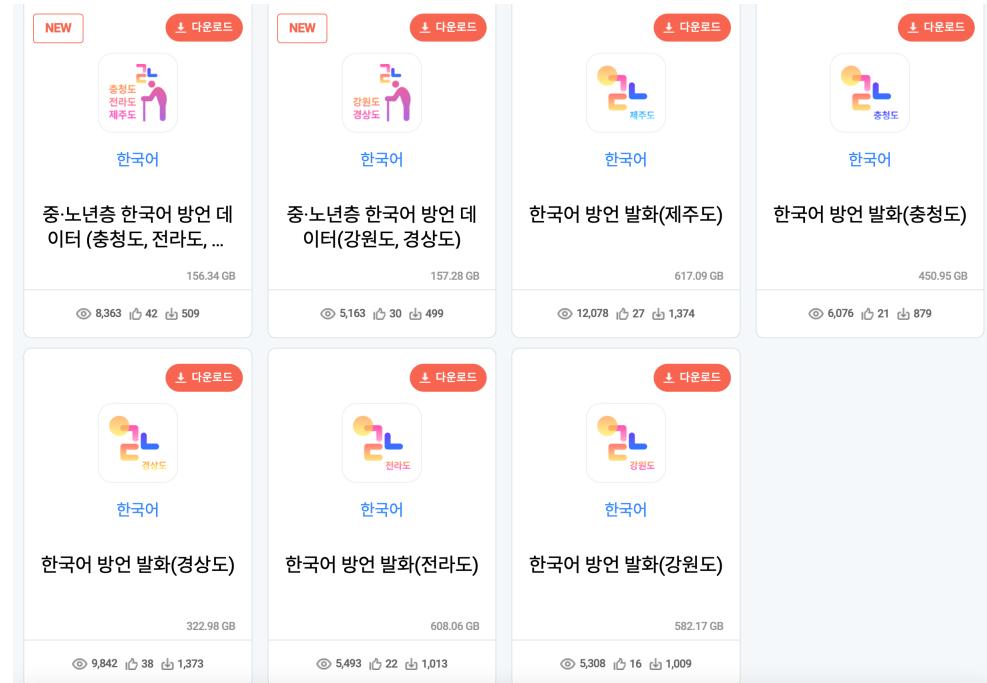
“**표준어 질문 (source) - 사투리 대답 (target)**
의 데이터셋으로 챗봇 모델을 학습시킨다.”

“**표준어 질문 (source) - 표준어 대답 (target)**
의 데이터셋으로 챗봇 모델을 학습시킨 뒤, 번역
모델을 이용해 **표준어 대답을 사투리로 번역한다.**”

02. 데이터 수집/전처리

01. 방언 데이터

1. 한국어 방언 발화 데이터



- 경상도 / 제주도 / 충청도 / 전라도 / 강원도
- n 명의 사람이 대화하는 음성 파일을 텍스트로 변환시킨 데이터셋
- json 형태로 되어있음
- 각 문장에 대해 발화자 Id, 표준어, 방언, 어절 별 방언인지 여부 등의 칼럼으로 구성

form	standard_form	dialect_form	speaker_id	start	end	note	eojeolList	Python
자 음식을 멀 좋아하느냐 하면은	자 음식을 멀 좋아하느냐 하면은	자 음식을 멀 좋아하느냐 하면은	1	0.01	3.45		[{'eojeol': '자', 'id': 1, 'isDialect': False, ...}	
나는 사실은 다 좋아합니다 다 좋아하는데 특히 육류 돼지고기	나는 사실은 다 좋아합니다 다 좋아하는데 특히 육류 돼지고기	나는 사실은 다 좋아합니다 다 좋아하는데 특히 육류 돼지고기	1	3.45	8.53		[{'eojeol': '나는', 'id': 1, 'isDialect': False, ...}	
아 자기 아 오리를 싫어했었구나	아 자기 아 오리를 싫어했었구나	아 자기 아 오리를 싫어했었구나	2	8.53	12.53		[{'eojeol': '아', 'id': 1, 'isDialect': False, ...}	
머 못 (목)/(먹) (그라고)/(그리고) 모든걸 다 좋아하는 데 즐기진 않는다.	머 못 먹 그리고 모든걸 다 좋아하는 데 즐기진 않는다.	머 못 먹 그리고 모든걸 다 좋아하는 데 즐기진 않는다.	1	12.53	17.86		[{'eojeol': '머', 'id': 1, 'isDialect': False, ...}	
즐기진 않는다 자기는 나하고 반대네 나는	즐기진 않는다 자기는 나하고 반대네 나는	즐기진 않는다 자기는 나하고 반대네 나는	2	17.86	21.54		[{'eojeol': '즐기진', 'id': 1, 'isDialect': False...}	
야채를 좋아하거든.	야채를 좋아하거든.	야채를 좋아하거든.	2	21.54	23.50		[{'eojeol': '야채를', 'id': 1, 'isDialect': False...}	
음~ 야채	음~ 야채	음~ 야채	1	23.50	25.91		[{'eojeol': '음~', 'id': 1, 'isDialect': False, ...}	
에피사이드 에피타이저	에피사이드 에피타이저	에피사이드 에피타이저	2	25.91	27.84		[{'eojeol': '에피사이드', 'id': 1, 'isDialect': Fal...}	
머라노 그 곁들어 먹는	머라노 그 곁들어 먹는	머라노 그 곁들어 먹는	1	27.84	30.76		[{'eojeol': '머라노', 'id': 1, 'isDialect': False...}	
곁들어먹는거 근데 야채를 많이 먹어야 건강해요.	곁들어먹는거 근데 야채를 많이 먹어야 건강해요.	곁들어먹는거 근데 야채를 많이 먹어야 건강해요.	2	30.76	36.11		[{'eojeol': '곁들어먹는거', 'id': 1, 'isDialect': Fa...}	

각 지역별 한국어 발화

- 제주도: 약 277만 row
- 전라도: 약 216만 row
- 경상도: 약 216만 row
- 충청도: 약 10만 row
- 강원도: 약 157만 row

각 지역별 중노년층 한국어 대화 데이터

- 전라도: 약 6만 row
- 경상도: 약 7만 row

01. 방언 데이터

1. 한국어 방언 발화 데이터

1-1. 챗봇 학습을 위해 대화 형식으로 가공

	speaker_1	speaker_1_eojeol_sum	speaker_2	speaker_2_eojeol_sum
0	자 음식을 멀 좋아하느냐 하면은 나는 사실은 다 좋아합니다 다 좋아하는데 특히 육류...	0	아 자기 아 오리를 싫어했었구나	0
1	머 못 둑 그리고 모든걸 다 좋아하는 데 즐기진 않는다.	2	즐기진 않는다 자기는 나하고 반대네 나는 야채를 좋아하거든.	0
2	음~ 야채	0	에피사이드 에피타이저	0
3	머라노 그 걸들어 먹는	0	곁들어먹는거 근데 야채를 많이 먹어야 건강해요.	0
4	그 당연한거지.	0	아니 왜 그렇다면 내가 과학적으로 말해볼께. 이 되지고나 소고기 이런거 육류는 사실...	3
5	그래서 음식에 가린다는 것은 나는 한편으로 불행하다 생각합니다. 아니 어쩔수 없이 ...	1	그러면 사실은 우리는 뭐 없어서 못 먹지.	1
6	그러니까	0	자기는 자기는 특히 더 그렇고 아무거나 뭐 다 감사하게 잘 먹는데 형부같은 경우가 ...	2
7	밥을 쟁겨먹는 생각도 의미 없다고 생각해. 영양분이 되고 배가 부르면은 끄 키식사...	1	근데 나는 그기서 꽈금 반대하는거 머니면 이 이제 밥이 탄수화물이잖아 탄수화물인데 ...	2
8	전부 다 지방으로 바뀐다매	1	아~ 지방으로 바뀌는게 아니고 탄수화물 지방 상대의 그~ 이게 있잖아 탄수화물 지방...	5
9	그 담에 미네랄 그렇구나 아 그래서 요새 우리 아침을 계속 쟁겨먹구나.	0	근데 내가 나는 직장생활 하면서 근 십년동안 아침을 안 먹었던 키 안 큰거는 옛날...	4

- 발화자 수가 2인 데이터만 필터링한 뒤,
speaker 1 (질문) 과 speaker 2 (대답) 칼럼
형태로 구성

1-2. 번역기 학습을 위해 한 문장에 대해 표준어 - 방언 쌍 형태로 가공

standard_form	dialect_form
아니면 다 글지 뭐 잡채 아니면 뭐	아니면 다 글지 뭐 잡채 아니면 뭐
전쟁 영화예요 전쟁 영화 그~ 빅뱅에 누구 나았다가는 최근에 알았는데	전쟁 영화예요 전쟁 영화 그~ 빅뱅에 누구 나았다가는 최근에 알았는데
또 눈꽃 축제 같은 경우에도 겨울에만 즐길 수 있는 행사다 보니까 올해에	또 눈꽃 축제 같은 경우에도 겨울에만 즐길 수 있는 행사다 보니까 올해에
다 물어뜯고 화장지는 화장지를 펼쳐가지고 난장판을 쳐놓고	다 물어뜯고 화장지는 화장지를 펼쳐가지고 난장판을 쳐놓고
그래서 요새는 이제 화순 주변에 그 무등산도 있고 어 어 안양산도 있고 뭐 백아산도...	그래서 요새는 이제 화순 주변에 그 무등산도 있고 어 어 안양산도 있고 뭐 백아산도...
...	...
뭐가 중요한데	뭣이 중요하니
너땜에	너땜시
그래가지고	그래가고
그래가지고 되겠나?	그래가고 되겠나?
그래가지고 되겠나?	그래가지고 되겠나?

01. 대화 데이터

2. 일상 대화 데이터

	req	res
0	너 좋아하는 차 종류 있어?	무슨 차? 자동차? 마시는 차?
1	마시는 차 말한 거야	나 둥글레, 옥수수, 보리차 좋아해
2	완전 곡물류 좋아하네	끓이기 귀찮아서 냉침해 먹어
3	그럼 오래 걸리지 않아?	끓이는 것보다는 훨씬 오래 걸리지
4	근데 냉침 하는 것도 귀찮겠다	응 그래서 매일은 안 먹고 가끔 마셔
5	그럼 엄청 귀찮지는 않겠네?	그치 매일 마시면 매일 해야 되잖아
6	생각해 보니깐 그렇긴 하네	언니는 무슨 차 좋아하는데?
7	나는 밀크티도 좋아하고 루이보스도 좋아해	고급져 나 페퍼민트도 좋아한다
8	너 구해줘 홈즈 봐?	매번 보는 건 아니고 가끔 와이?
9	나 지금 보는 중인데 서울은 진짜 비싸다 싶어서	그치 근데 나오는 사람들 다 부자인가 봐
10	그니깐 웬만하면 다 돈 있더라	내말이 경제적으로 힘들다더니 다 거짓말
11	인정 나만 힘든 거였어	아냐 나도 힘들어 나도 추가해줘
12	진짜 어디 무인도 들어가서 살아야 하나 봐	땅값은 싸겠지만 무서울 듯
13	정부는 대체 무엇을 하는 겁니까	맞아요 부동산 정책 과연 실행할 수 있는 겁니까?
14	열심히 벌어서 내 집 마련을 꿈꿉니다	저도 그렇습니다 내 집 마련 완전 소망
15	나만 쓸 수 있는 나만의 공간	완전 생각만 해도 행복하다
16	대구랑 광주랑 열차 생기는 거 알아?	응응 봤어 얼마나 빠를까?
17	글쎄? 그래도 두 시간 정도는 걸리지 않을까?	기차 타고 가본 적이 없어서 나도 잘 모르겠다
18	개통하면 타 볼 의향이 있어?	나 태어나서 대구 한번 밖에 안 가봤어
19	생각보다 많이 안 가봤네?	그치 그래서 개통하면 한번 가볼래
20	다른 지역으로 가는 것도 많이 생기면 좋겠다	응 부산으로 가는 기차 없는 거 알았어?

	Q	A	label
0	12시 땡!	하루가 또 가네요.	0
1	1지망 학교 떨어졌어	위로해 드립니다.	0
2	3박4일 놀러가고 싶다	여행은 언제나 좋죠.	0
3	3박4일 정도 놀러가고 싶다	여행은 언제나 좋죠.	0
4	PPL 심하네	눈살이 찌푸려지죠.	0
5	SD카드 망가졌어	다시 새로 사는 게 마음 편해요.	0
6	SD카드 안돼	다시 새로 사는 게 마음 편해요.	0
7	SNS 맞팔 왜 안하지ㅠㅠ	잘 모르고 있을 수도 있어요.	0
8	SNS 시간낭비인 거 아는데 매일 하는 중	시간을 정하고 해보세요.	0
9	SNS 시간낭비인데 자꾸 보게됨	시간을 정하고 해보세요.	0
10	SNS보면 나만 빼고 다 행복해보여	자랑하는 자리니까요.	0
11	가끔 궁금해	그 사람도 그럴 거예요.	0
12	가끔 뭐하는지 궁금해	그 사람도 그럴 거예요.	0
13	가끔은 혼자인게 좋다	혼자를 즐기세요.	0
14	가난한 자의 설움	돈은 다시 들어올 거예요.	0
15	가만 있어도 땀난다	땀을 식혀주세요.	0
16	가상화폐 풀딱 망함	어서 잊고 새출발 하세요.	0
17	가스불 켜고 나갔어	빨리 집에 돌아가서 끄고 나오세요.	0
18	가스불 켜놓고 나온거 같아	빨리 집에 돌아가서 끄고 나오세요.	0
19	가스비 너무 많이 나왔다.	다음 달에는 더 절약해봐요.	0

- AI Hub / 타 챗봇 프로젝트의 데이터를 수집해 일상적인 대화 형태의 데이터셋 수집 및 가공

03. Model

- case 1) 표준어 질문: 사투리 대답

01. Transformer model

1. Transformer 모델 구현

- 타 프로젝트의 모델 직접 구현 코드 / Attention is all you need 논문 구현 코드 등을 참고해 basic 한 Transformer 모델을 구현
- 표준어 질문 : 사투리 대답 의 질의응답 쌍으로 학습시킨다면 번역된 답변과 정상적 대화가 가능할 것이라 예측

2. Transformer 모델 학습

- 20 epoch / batch size 64 / tokenizer 띄워쓰기 / vocab_size (30만개)

3. 학습 결과

```

 전체 소스 토큰: ['<sos>', '오늘', '저녁으로', '뭐먹을까?', '<eos>']
 소스 문장 인덱스: [2, 523, 17150, 0, 3]
 대답: ['근데', '그', '하고', '보고', '뭘', '국내', '오히려', '다녔는데', '그', '뭘', '끼니로도', '이렇게', '딱히', '공과금이라']

 전체 소스 토큰: ['<sos>', '어제', '저녁에는', '뭐', '했어?', '<eos>']
 소스 문장 인덱스: [2, 1067, 7614, 5, 0, 3]
 대답: ['때', '근데', '느셔가지고', '이렇게', '무료하니', '<eos>']

 전체 소스 토큰: ['<sos>', '아침에', '일찍', '일어나는', '건', '피곤한', '일이지?', '<eos>']
 소스 문장 인덱스: [2, 684, 1158, 2958, 97, 12561, 0, 3]
 대답: ['때', '근데', '느셔가지고', '이렇게', '무료하니', '<eos>']

```

- 문맥이 전혀 이해가지 않을 토큰들이 예측되었으며, 올바른 문장 형태를 유지하지 못함
- 가공한 데이터셋의 한계 발견 : 질의를 염두에 두고 만들어진 데이터셋이 아님
- 어느 정도 사전학습이 된 모델을 Task 에 맞게 Fine tuning 시키는 것이 방향성에 적합

```

class Transformer(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = Encoder()
        self.decoder = Decoder()

    def forward(self, src, tgt, src_mask, tgt_mask):
        enc_src = self.encoder(src, src_mask)
        dec_src = self.decoder(tgt, enc_src, src_mask, tgt_mask)
        return dec_src

```

```

class Encoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.tok_embedding = Embedding(375654, 256)
        self.pos_embedding = Embedding(100, 256)
        self.layers = ModuleList([EncoderLayer() for _ in range(6)])
        self.norm = LayerNorm(256)

    def forward(self, src, src_mask):
        x = self.tok_embedding(src) + self.pos_embedding(torch.arange(src.size(1), device=src.device))
        for layer in self.layers:
            x = layer(x, src_mask)
        return self.norm(x)

```

```

class Decoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.tok_embedding = Embedding(387764, 256)
        self.pos_embedding = Embedding(100, 256)
        self.layers = ModuleList([DecoderLayer() for _ in range(6)])
        self.norm = LayerNorm(256)
        self.fc_out = Linear(in_features=256, out_features=387764, bias=True)

    def forward(self, tgt, enc_src, src_mask, tgt_mask):
        x = self.tok_embedding(tgt) + self.pos_embedding(torch.arange(tgt.size(1), device=tgt.device))
        for layer in self.layers:
            x = layer(x, enc_src, src_mask, tgt_mask)
        x = self.norm(x)
        x = self.fc_out(x)
        return x

```

```

class EncoderLayer(nn.Module):
    def __init__(self):
        super().__init__()
        self.self_attn_norm = LayerNorm(256, eps=1e-05, elementwise_affine=True)
        self.self_attention = MultiHeadAttentionLayer(4, 256)
        self.self_attn_layer_norm = LayerNorm(256, eps=1e-05, elementwise_affine=True)
        self.ff_norm = PositionwiseFeedforwardLayer(256, 512)
        self.dropout1 = Dropout(p=0.1, inplace=False)
        self.dropout2 = Dropout(p=0.1, inplace=False)
        self.dropout3 = Dropout(p=0.1, inplace=False)

    def forward(self, x, src_mask):
        x = self.self_attn_norm(x)
        x = self.self_attention(x, x, x)
        x = self.dropout1(x)
        x = self.self_attn_layer_norm(x)
        x = self.ff_norm(x)
        x = self.dropout2(x)
        x = self.dropout3(x)
        return x

```

```

class DecoderLayer(nn.Module):
    def __init__(self):
        super().__init__()
        self.enc_attn_norm = LayerNorm(256, eps=1e-05, elementwise_affine=True)
        self.enc_attention = MultiHeadAttentionLayer(4, 256)
        self.enc_attn_layer_norm = LayerNorm(256, eps=1e-05, elementwise_affine=True)
        self.self_attn_norm = LayerNorm(256, eps=1e-05, elementwise_affine=True)
        self.self_attention = MultiHeadAttentionLayer(4, 256)
        self.self_attn_layer_norm = LayerNorm(256, eps=1e-05, elementwise_affine=True)
        self.ff_norm = PositionwiseFeedforwardLayer(256, 512)
        self.dropout1 = Dropout(p=0.1, inplace=False)
        self.dropout2 = Dropout(p=0.1, inplace=False)
        self.dropout3 = Dropout(p=0.1, inplace=False)
        self.dropout4 = Dropout(p=0.1, inplace=False)

    def forward(self, tgt, enc_src, src_mask, tgt_mask):
        tgt = self.self_attn_norm(tgt)
        tgt = self.self_attention(tgt, tgt, tgt)
        tgt = self.dropout1(tgt)
        tgt = self.self_attn_layer_norm(tgt)
        tgt = self.enc_attention(tgt, enc_src, enc_src)
        tgt = self.dropout2(tgt)
        tgt = self.enc_attn_layer_norm(tgt)
        tgt = self.ff_norm(tgt)
        tgt = self.dropout3(tgt)
        tgt = self.dropout4(tgt)
        return tgt

```

02. Pretrained models

1. T5 / GPT2 기반의 : LLM 모델

- 해당 모델을 기반으로 하면서 대용량의 한국어 말뭉치로 사전학습된 모델 선정
- 참고할 레퍼런스가 충분하며, 어느 정도 성능이 검증된 모델 선정

The screenshot shows the Hugging Face Model Hub interface. At the top, there's a search bar with 'paust' and a dropdown menu with 'paust/pko-t5-base'. Below the search bar are various filters: Text2Text Generation, Transformers, PyTorch, Korean, t5, text-generation-inference, Inference Endpoints, arxiv:2105.09680, and License: cc-by-4.0. The main card for 'pko-t5-base' is displayed, showing its name, a purple waveform icon, and a download count of 1,252 last month. It includes sections for 'Model card', 'Files', 'Community', 'Edit model card', 'Inference API', 'Model tree for paust/pko-t5-base', and 'Finetunes'. The 'Inference API' section notes that the model does not have enough activity to be deployed yet.

KoGPT2 (한국어 GPT-2) Ver 2.0

GPT-2는 주어진 텍스트의 다음 단어를 잘 예측할 수 있도록 학습된 언어모델이며 문장 생성에 최적화 되어 있습니다. KoGPT2는 부족한 한국어 성능을 극복하기 위해 40GB 이상의 텍스트로 학습된 한국어 디코더(decoder) 언어모델입니다.

The screenshot shows a demo interface for KoGPT2. It displays a sequence of Korean words: '아버지가 방에 들어가신다.' followed by '</s>'. Below this, a yellow button contains the text 'Autoregressive Decoder'. At the bottom, it shows another sequence: '<s> 아버지가 방에 들어가신다.'

<https://paust-studio.notion.site/pre-trained-T5-7072a7049e184eb6998b04e588b226e0>
<https://huggingface.co/paust/pko-t5-base>
<https://github.com/SKT-AI/KoGPT2>

02. Pretrained models

2. paust/t5bse model & skt/kogpt2 학습 결과

In [13]:

```
# 문장 생성 3 - model 2
generated_ids = model2.generate(
    input_ids,
    do_sample=True,
    max_length=128,
    top_k=0,
    top_p=0.90,
    num_return_sequences=3
)
```

```
# 생성된 토큰을 텍스트로 변환
generated_text = tokenizer.decode(generated_ids[0], skip_special_tokens=True)
print("Generated text:", generated_text)
```

Generated text: 오늘 저녁은 뭘 먹을까? 그래도 좋은 건 역시 고기류가 맛있으니까 오늘 저녁 뭘 먹을까 가성비가 좋은 쪽갈비 종류별로 몇 개 골라보고 또 오늘 저녁에 뭘 먹지 모르겠다 그래도 오늘 저녁에 뭘 먹지 몰라서 고민하다가 결국엔 상어콩 등갈비찜으로 결정했다 근데 생각보다 양이 많아서 놀랬다 그래 오늘 저녁 뭘 먹지 몰라서 고민하다가 결국 내일 먹을

Deocoder 부분의
generate 메서드 옵션 조절

Output:

```
-----
0: 오늘 저녁은 뭘 먹을까? 저는 오늘 아침에도 안 먹고 일단 오늘 저녁도 안 먹고 오늘 또 일찍 일어나니까 내일은 또 또 안 먹고 롯데에서 사 먹는 거 먹고 싶어요 오늘 점심은 뭘 먹냐고 뭘 먹냐고 그 롯데가 그 롯데에서 사 먹는 거예요 그거는 뭐고 그기 신라면 거기 먹어 봤는데 그기 신라면 거기 먹어 봤고 신라면 거기 파는 괴당지하고 그기 소고기 먹어 봤는데 맛있
```

```
1: 오늘 저녁은 뭘 먹을까? 야 근데 우리 집이 뭐냐면 라면이 아니잖아 근데 라면 말고 그냥 그 쎈 거 뭐라 해야 되지 그거 뭐꼬 그냥 쎈 거 그런 거 말고 그거 해물 쎈 거 원 거고 약간 매워야 된다매 매운탕이 그 뭐고 매운탕은 매운탕도 매운탕도 아니고 그냥 뭐 그 뭐꼬 그냥 쎈 거 그런 거 말고 그냥 그냥 쎈 거는 뭐 매운탕도 아니고 그냥 그거 매운탕도 아니고 쎈 거는 매운탕으로 알고 있는데 니가 말한 대로 라
```

```
2: 오늘 저녁은 뭘 먹을까? 내일은 꼭 챙겨 먹어야겠다 이게 또 어제도 뉴스도 다 챙겨지고 오늘도 뉴스도 다 챙겨지는 그건 그렇네 오늘 저녁은 뭘 먹고 싶어
```

```
Generated text: 어제 저녁을 먹으면서 언니가 언니야 고마워 고마워 뭘 먹지 고민을 하는데 우리는 우리는 저녁을 맛있게 먹고 오는 거지 오늘은 뭘 먹을까 고민을 하다가 새우 튀김을 먹을까 고민을 하는 거야 근데 고민하는 이유가 새우튀김이 맛있어 보여 그래 그래 오늘은 새우튀김을 해 먹을까 해서 고민을 하는 거야 근데 내가 이미 주변에 새우튀김 가게가 몇
```

- 각각 5 epoch / 문장 길이 필터링 / 일정 개수의 방언 포함 / generate parameter 조절 등 구현
- 문장의 생성은 잘 되지만, 사투리 반영이 잘 되지 않는 모습을 보임
- 학습시킨 문장의 길이가 길어서인지, 답변 생성 또한 길이가 상당히 길었음

03. Model

- case 2) 표준어 질문 : 표준어 대답

01. 표준어 챗봇 구현

1. 표준어 질문 → 표준어 대답 (1차 시도)

	speaker_1	speaker_2	id
0	자 음식을 멀 좋아하느냐 하면은 나는 사실은 다 좋아합니다 다 좋아하는데 특히 육류...	아 자기 아 오리를 싫어했었구나	DKCI20000001
1	머 못 묵 그라고 모든걸 다 좋아하는 데 즐기진 않는 다.	즐기진 않는다 자기는 나하고 반대네 나는 야채를 좋아하거든.	DKCI20000001
2	음~ 야채	에피사이드 에피타이저	DKCI20000001
3	머라노 그 곁들어 먹는	곁들어먹는거 근데 야채를 많이 먹어야 건강해요.	DKCI20000001
4	그 당연한거지.	아니 왜 그렇냐면 내가 과학적으로 말해볼께. 이 돼지고나 소고기 이런거 육류는 사실...	DKCI20000001
...
113814	요즘에 트로트가 진짜 많이 이게 대세잖아요. 방송 막 틀어보면 뭐~ 방송 상사 뭐~...	아~ 내가 임영웅을 알았는 거는 미스터 트롯 그 프로를 보면서 알았는데 야가 이 사...	DKSR20007343
113815	별로 저는 그거 노래 그렇게 막 잘하 아~ 잘하는 건 알겠는데 근데 막 계속 보고 ...	내가 그 당시에는 임영웅 영웅이한테 빠져 갖고 다른 사람이 안 보여 가주고 몰랐는데...	DKSR20007343
113816	근데 나는 엄마가 그렇게 누군가를 좋아하고 관심있게 보고 뭐~ 이러는 게 진짜 쯤 ...	배우 중에는 내가 영하고 드라마다 이런 거를 잘 안 보기 때문에 나는 좋아하는 사람...	DKSR20007343
113817	나는 요즘에는 유아인 예전에 그~ 사도나 배태랑 그 때 이천십오 년도에 개봉했을 때 ...	개그맨은 옛날에는 우리가 그거 저기 개그 프로 근데 뭐~ 제목은 내가 모르겠는데 뭐...	DKSR20007343
113818	엄마는 그러면 놀면 어때라는 그~ 아이 아~ 맞지 놀면 어때 맞지 -유- 유재석이 ...	아~ 나는 유재석이라카는 그 사람 이미지가 매 아래 친카 칙한 걸로 이르케 보여주고...	DKSR20007343

AI 허브 한국어 방언 발화

- T5-model에 fine-tuning
- 출력 결과

발화 음성 데이터의 특성상 문장이 길고 매끄럽지 않으며, 감탄사 등이 연속해서 나오는 경우가 많음

You: 안녕?
 Chatbot: 비밀비재하니?
 You: 너 무슨 말이야?
 Chatbot: 어 그거 뭐 진짜 잘 안 안 하잖아.

01. 표준어 챗봇 구현

2. 표준어 질문 → 표준어 대답 (2차 시도)

	req	res
0	너 좋아하는 차 종류 있어?	무슨 차? 자동차? 마시는 차?
1	마시는 차 말한 거야	나 둥글레, 옥수수, 보리차 좋아해
2	완전 곡물류 좋아하네	끓이기 귀찮아서 냉침해 먹어
3	그럼 오래 걸리지 않아?	끓이는 것보다는 훨씬 오래 걸리지
4	근데 냉침 하는 것도 귀찮겠다	응 그래서 매일은 안 먹고 가끔 마셔
5	그럼 엄청 귀찮지는 않겠네?	그치 매일 마시면 매일 해야 되잖아
6	생각해 보니깐 그렇긴 하네	언니는 무슨 차 좋아하는데?
7	나는 밀크티도 좋아하고 루이보스도 좋아해	고급져 나 페퍼민트도 좋아한다
8	너 구해줘 홈즈 봐?	매번 보는 건 아니고 가끔 와이?
9	나 지금 보는 중인데 서울은 진짜 비싸다 싶어서	그치 근데 나오는 사람들 다 부자인가 봐
10	그니깐 웬만하면 다 돈 있더라	내말이 경제적으로 힘들다더니 다 거짓말
11	인정 나만 힘든 거였어	아냐 나도 힘들어 나도 추가해줘
12	진짜 어디 무인도 들어가서 살아야 하나 봐	땅값은 싸겠지만 무서울 듯
13	정부는 대체 무엇을 하는 겁니까	맞아요 부동산 정책 과연 실행할 수 있는 겁니까?
14	열심히 벌어서 내 집 마련을 꿈꿉니다	저도 그렇습니다 내 집 마련 완전 소망
15	나만 쓸 수 있는 나만의 공간	완전 생각만 해도 행복하다
16	대구랑 광주랑 열차 생기는 거 알아?	응응 봤어 얼마나 빠를까?
17	글쎄? 그래도 두 시간 정도는 걸리지 않을까?	기차 타고 가본 적이 없어서 나도 잘 모르겠다
18	개통하면 타 볼 의향이 있어?	나 태어나서 대구 한번 밖에 안 가봤어
19	생각보다 많이 안 가봤네?	그치 그래서 개통하면 한번 가볼래
20	다른 지역으로 가는 것도 많이 생기면 좋겠다	응 부산으로 가는 기차 없는 거 알았어?

- AI 허브 '주제별 텍스트 일상 대화 데이터'로 데이터셋 대체
- T5 model 대신 Hugging Face의 사전학습된 KoGPT2 model과 tokenizer 활용

01. 표준어 챗봇 구현

2. 표준어 질문 → 표준어 대답 (2차 시도)

```

def preprocess(input_string):
    cleaned1 = re.sub(r'(~|#)(=\$)|(?<=\$)(~|#)', '', input_string)

    cleaned2 = re.sub(r'&+\w+&|@+\w+&|@\w+', '<고유명사>', cleaned1)

    cleaned3 = re.sub(r'\S*(\(\w*\))\S*', '', cleaned2)

    # Excluding <unk>
    special = re.compile(r'(^가-힣\s\d.,?,<알수없음>) ')
    cleaned4 = special.sub('', cleaned3)

    # 특정 단어 제거
    words_to_remove = ['아', '그', '음', '오', '학', '에', '아하', '엥', '흑', '힐', '아아', '후', '흘', '웅', '마자', '아아아', '아아아아',
    '햇', '앗', '어', '헉', '휴', '흡', '하', '야쓰', '열', '꿔꿔', '키끼', '글구', '마즈마즈', '쏴리', '흐흐']
    pattern = r'\b(' + '|'.join(map(re.escape, words_to_remove)) + r')~?\b'
    cleaned5 = re.sub(pattern, '', cleaned4)

    cleaned6 = re.sub(r'*.*?*.*', '', cleaned5)

    cleaned_text_final = re.sub(r"^[^가-힣0-9a-zA-Z\s.,!?.]", "", cleaned6)

    cleaned_text_final = re.sub(r'\s+', ' ', cleaned_text_final).strip()

    return cleaned_text_final

# 한 글자 이하로 되어 있는 모든 행 제거
all_data = all_data[~all_data['req'].str.len().le(1) & ~all_data['res'].str.len().le(1)]

# 100자 이상인 행 모두 제거
all_data = all_data[~all_data['req'].str.len().gt(100) & ~all_data['res'].str.len().gt(100)]

```

- KoGPT2 사전학습 모델 불러오기
- Dataloader 구성

← 전처리

- 찾은 규칙을 기반으로 텍스트 정제
- 길이가 너무 짧거나, 너무 긴 행 모두 제거

```

1 koGPT2_TOKENIZER = PreTrainedTokenizerFast.from_pretrained("skt/kogpt2-base-v2",
2                         bos_token=BOS, eos_token=EOS, unk_token='<unk>',
3                         pad_token=PAD, mask_token=MASK)
4 model = GPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2')
5
train_set = ChatbotDataset(all_data, max_len=40)
train_dataloader = DataLoader(train_set,
                                batch_size=32,
                                num_workers=0,
                                shuffle=True,
                                collate_fn=collate_batch,
                                )

```

01. 표준어 챗봇 구현

2. 표준어 질문 → 표준어 대답 (2차 시도)

다음과 같이 학습

- Epoch : 10
- Learning rate : 3e-5
- Loss : CrossEntropyLoss
- Optimizer : Adam optimizer

출력 예시 >

```

user > 저녁 메뉴 추천해줘
Chatbot > 나는 오늘 백반 집밥 어때
user > 나는 점심 부대찌개 먹었어
Chatbot > 점심 부대찌개는 맛있지
user > 영화 뭐 볼까?
Chatbot > 나는 최근에 개봉한거 보고싶더라

user > 지금 몇시야?
Chatbot > 9시50분
user > 지금 날씨 어때?
Chatbot > 지금 추위
user > 지금 뭐해?
Chatbot > 나 지금 밥먹을까 생각중이야

user > 어떤 음식 제일 좋아하세요?
Chatbot > 요즘은 햄버거 좋아해요
user > 혹시 고기도 좋아하시나요?
Chatbot > 그럼요 고기는 좋아하죠

```

```

for epoch in range(num_epochs):
    print(f"Epoch {epoch + 1}/{num_epochs}")
    for batch_idx, samples in enumerate(tqdm(train_dataloader, desc=f"Training Epoch {epoch + 1}/{num_epochs}")):
        optimizer.zero_grad()
        token_ids, mask, label = samples
        token_ids = token_ids.to(device)
        mask = mask.to(device)
        label = label.to(device)
        out = model(token_ids)
        out = out.logits      #Returns a new tensor with the logit of the elements of input
        mask_3d = mask.unsqueeze(dim=2).repeat_interleave(repeats=out.shape[2], dim=2)
        mask_out = torch.where(mask_3d == 1, out, Sneg * torch.ones_like(out))
        loss = criterion(mask_out.transpose(2, 1), label)
        avg_loss = loss.sum() / mask.sum()
        avg_loss.backward()
        optimizer.step()

```

- 대부분 입력값에 어울리는 대답을 출력함
- 학습 데이터를 그대로 출력하는 것이 아니라 답변을 새로 생성해냄

[개선점]

- 두 번 이상의 대화가 불가능하다는 점
- 학습 데이터의 특성상 띄어쓰기가 제대로 되지 않을 수 있다는 점
- 가끔 매끄럽지 않은 답변을 할 때가 있음

03. Model

- case 2) 표준어 대답 사투리 번역

01. KoBART

1. KoBART 다국어 번역 모델 (1차 시도)

- 표준어 / 사투리를 각각 일종의 다른 언어로 간주
- 번역 Task에 특화된 M2M100, MarianMT 같은 다국어 번역 모델을 튜닝하는 것이 목적
- 토큰화 성능 저조 / 알 수 없는 언어로 의미 없는 문장 생성 등 여러 오류 발생

2. 한국어 특화된 사전학습 모델의 Conditional Generate 활용 (2차 시도)

- Conditional Generation : 큰틀에서 어순과 문법은 유지하되 특정 단어나 표현을 대체하는 것으로 간주하고 조건에 맞는 출력만 내보냄
- 문맥 / 어순 / 문법은 유지하되 복잡한 텍스트 변환이 요구되므로 양방향성 활용에 적합한 BART 모델 선정
- 5개 지역의 표준어 - 사투리 변환 모델을 KoBART 의 Conditional Generate 모델을 튜닝해 구축

3. Model Setting

- 별도의 config를 설정하지 않은 KoBART (gogamaza/kobart=base-v2) 모델 사용
- Optimizer : AdamW ($lr = 2e-5$)
- Epoch 5 / Batch Size 32

01. KoBART

4. 학습 결과 (제주도)

```
# 입력 문장
input_sentence = "아 그래? 그러니까 요새 다 힘든거라 골채 해서 이렇게 구멍 막아서"
# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to("cuda")
# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5)
# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

| 아 기? 그러니까 요새 다 힘든거라 골채 해그네 영 구멍 막아그네

```
# 입력 문장
input_sentence = "전화 오는 거 같은데"
# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to("cuda")
# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5)
# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

| 전화 오는 거 닮은디

```
# 입력 문장
input_sentence = "밥 먹었니?"
# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to("cuda")
# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5)
# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

| 밥 먹언?

- 먹었니 -> 먹언 / 같은데 -> 닮은디 등 각 표준어에 대해 정상적으로 번역

01. KoBART

4. 학습 결과 (전라도)

```
# 입력 문장
input_sentence = "나물에 밥 비며 먹고싶네"
# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt")
# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5)
# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

| 나물에 밥 비며 먹고잡네

```
# 입력 문장
input_sentence = "그런데 이게 제주도로까지 가게가 이전이 됐는데"
# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt")
# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5)
# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

| 근디 이게 제주도로까지 가게가 이전이 됐는디

```
Python ▾
# 입력 문장
input_sentence = "도영아 너때문에 살아"
# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt")
# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5)
# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

| 도영아 니땜시 살아야

- 됐는데 -> 됐는디 / 먹고싶네 -> 먹고잡네 등 각 표준어 정상 번역
- 외에 3개 지역 모두 정상적으로 번역되는 것 확인

03. Model - 학습 모델 결합

01. 학습 모델 결합

1. 모델 저장 및 로드

- 학습이 완료된 챗봇 / 번역 모델 가중치를 각각 json 형식으로 저장한 뒤 load
- 목적에 맞도록 각 모델 클래스 선언 후 질문 -> 답변 생성 -> 생성된 답변 번역 하도록 구현

```
1 with torch.no_grad():
2     while 1:
3         q = input("user > ").strip()
4         if q == "quit":
5             break
6         a = ""
7         while 1:
8             input_ids = torch.LongTensor(tokenizer_chatbot.encode(Q_TKN + q + SENT + A_TKN + a)).unsqueeze(dim=0)
9             model_chatbot_3 = model_chatbot_3.to('cpu')
10            pred = model_chatbot_3(input_ids)
11            pred = pred.logits
12            gen = tokenizer_chatbot_3.convert_ids_to_tokens(torch.argmax(pred, dim=-1).squeeze().numpy().tolist())[-1]
13            if gen == EOS:
14                break
15            a += gen.replace("_", " ")
16            input_ids = tokenizer_int_cc.encode(a, return_tensors = "pt").to(device)
17            # 모델 예측
18            output_ids = model_int_cc.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)
19            # 결과 해독
20            output_sentence = tokenizer_int_cc.decode(output_ids[0], skip_special_tokens=True)
21            print("Chatbot > {}".format(output_sentence.strip()))
```

01. 학습 모델 결합

2. 결과물

user > 요즘 날씨 어때?
Chatbot > 요즘 선선하니 딱 좋아너무 더워서 싫더라
user > 둘이 사귀면 잘 어울리겠다 으휴
Chatbot > 아주 한 쌍의 바퀴벌레가 탄생할뻔 했지 알바하면서 잘생긴 손님은 본 기억이 거의 없어너는 인
user > 난 커널스 카라멜이 젤 맛있어 아삭바삭해
Chatbot > 난 지금 배고파서 뭐든 입에 넣으면 맛있을 거 담아
user > 그래서 넌 언제 해
Chatbot > 나 무사 일년 전에 하지 않안
user > 밥 안 먹었더니 배고파
Chatbot > 야 너 너무 착하다난 절대 못 먹어

user > 밥 먹었니?
Chatbot > 응응 오늘 밥먹었어
user > 나 며칠 전에 독립하고 싶어서 집 알아보고 왔음
Chatbot > 독립하면 집도 엄청 비싸지 않아
user > 진짜 독립을 하려는 건 아니고
Chatbot > 근데 왜 부동산까지 가서 알아보는 거

01. 학습 모델 결합

2. 결과물

```
user > 밥 먹었니?  
Chatbot > 응응 오늘 밥먹언  
user > 뭐 먹었어?  
Chatbot > 나 햄버거 먹언  
user > 나는 뭐 먹었게?  
Chatbot > 나 햄버거
```

- 발화 의도에 맞는 답변을 성공적으로 생성하였음
- 답변 또한 정상적으로 번역되어 나오는 것을 확인할 수 있었음 (5개 지역 모두 테스트해봤으며, 가장 학습이 잘된 제주도 방언 위주로 첨부)

04. Result & Discussion

01. 의의 및 한계

1. 의의

- 프로젝트의 목적에 맞도록 두 가지 방향성을 설정하고, Task 별로 데이터 수집 ~ 모델 구현 및 파인튜닝 등의 파이프라인 경험
- 성능과 관계 없이, 처음 목표로 했던 생성형 모델 + 번역기 모델을 경험해보고, 성공적으로 결합
- 점점 소실되어가는 지역 방언을 주제로 언어 모델에 사투리를 접목시켜 이를 구사할 수 있는 언어모델을 다룬 것에서 가치가 있음

2. 한계

- 수집한 방언 데이터셋 자체가 음성 대화를 기반으로 텍스트 형태로 변환시킨 것이라 챗봇을 만들기엔 다소 깔끔한 데이터가 아니었음
- 특히 챗봇의 경우, 파인 튜닝을 위해 질문 -> 대답 형태의 데이터셋이 필요한데, 질적인 데이터셋을 다양하게 확보하는 것이 쉽지 않음
- GPU의 제한으로, 적은 수의 epoch 으로 학습 (여러 모델을 테스트하다보니 금방 닳는 문제 발생)

3. 개선방향

- 허깅페이스에 여러 NLP Task 를 위한 모델이 더 다양하게 존재하는데, 목적에 맞게 재검토하여 더 많은 모델 테스트 해볼 수 있음
- 방언이 강하게 들어간 문장들만 선별하여 번역기의 성능을 더욱 극단적으로 끌어올릴 수 있음
- 현재는 표준어-사투리 변환만 시도했지만, 반대로 사투리-표준어 변환도 필요함 (source / target 만 반대로 설정하면 구현 가능)

부록

01. 표준어-제주도

```
[ ] # 입력 문장  
input_sentence = "전화 오는 거 같은데"  
  
# 토큰화  
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 전화 오는 거 닮은디

01. 표준어-제주도

```
[ ] # 입력 문장  
input_sentence = "밥 먹었니?"  
  
# 토큰화  
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 밥 먹언?

01. 표준어-제주도

```
[ ] # 입력 문장  
input_sentence = "아 그래? 그러니까 요새 다 힘든거라 골채 해서 이렇게 구멍 막아서"  
  
# 토큰화  
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)  
# 모델 예측  
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens = True)  
print(output_sentence)
```

→ 아 기? 그러니까 요새 다 힘든거라 골채 해그네 영 구멍 막아그네

02. 표준어-전라도

```
[19] # 입력 문장
      input_sentence = "도영아 너때문에 살아"

      # 토큰화
      input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)

      # 모델 예측
      output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)

      # 결과 해독
      output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
      print(output_sentence)
```

→ 도영아 니땜시 살아야

02. 표준어-전라도

```
[35] # 입력 문장
      input_sentence = "그런데 이게 제주도로까지 가게가 이전이 됐는데"

      # 토큰화
      input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)

      # 모델 예측
      output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)

      # 결과 해독
      output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
      print(output_sentence)
```

→ 근디 이게 제주도로까지 가게가 이전이 됐는디

02. 표준어-전라도

```
[31] # 입력 문장
    input_sentence = "나물에 밥 비며 먹고싶네"

    # 토큰화
    input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)

    # 모델 예측
    output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)

    # 결과 해독
    output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
    print(output_sentence)
```

→ 나물에 밥 비며 먹고잡네

03. 표준어-경상도

```
[ ] # 입력 문장  
input_sentence = "너 영남 알프스 들어봤어?"  
  
# 토큰화  
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 니 영남 알프스 들어봤나?

03. 표준어-경상도

```
[ ] # 입력 문장  
input_sentence = "야 너 뭐 되니? 자신있니?"  
  
# 토큰화  
input_ids = tokenizer_int.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model_int.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer_int.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 야 니 뭐 되나? 자신있나?

03. 표준어-경상도

```
[ ] # 입력 문장  
input_sentence = "조금 배고프지 않니? 밥 먹었어?"  
  
# 토큰화  
input_ids = tokenizer_int.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model_int.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer_int.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 쪽 배고프다 아이가? 밥 먹었어?

04. 표준어-충청도

```
[ ] # 입력 문장  
input_sentence = "너무 좀 더럽고 그래 가지고"  
  
# 토큰화  
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 너무 좀 드럽고 그래 가주고

04. 표준어-충청도

```
[ ] # 입력 문장  
input_sentence = "이제 그런 걸 나눠 먹을 수가 없는 거야"  
  
# 토큰화  
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 이제 그런 걸 나눠 먹을 수가 없는 거.

04. 표준어-충청도

```
[ ] # 입력 문장  
input_sentence = "다른 거 관련된 걸 하나 시키고 싶은데"  
  
# 토큰화  
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)  
  
# 모델 예측  
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)  
  
# 결과 해독  
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)  
print(output_sentence)
```

→ 따른 거 관련된 걸 하나 시키고 싶은디

05. 표준어-강원도

```
[24] # 입력 문장
input_sentence = "정말? 음~ 나는 어머니 그다음에 나까지 해서"

# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)

# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)

# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

→ 증말? 음~ 나는 어멍이 그다음에 나까지 해서

05. 표준어-강원도

```
[44] # 입력 문장
input_sentence = "Passion Connected, 하나된 열정, 평창"

# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)

# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)

# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

→ Passion Connected, 하나된 열정, 평창

05. 표준어-강원도

```
[41] # 입력 문장
input_sentence = "내가 뭘 그렇게 잘못했어요?"

# 토큰화
input_ids = tokenizer.encode(input_sentence, return_tensors = "pt").to(device)

# 모델 예측
output_ids = model.generate(input_ids, max_length = 50, num_beams = 5, early_stopping = True)

# 결과 해독
output_sentence = tokenizer.decode(output_ids[0], skip_special_tokens=True)
print(output_sentence)
```

→ 내가 뭘 그렇게 잔모했어요?



Thank You