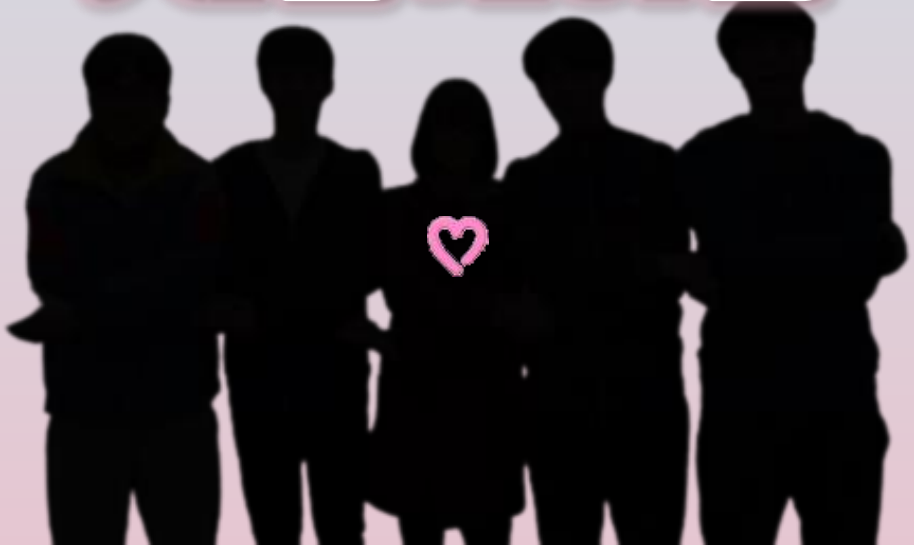
두근두근 쌍문동! "누가 그래? 우리가 친구라고."

쌍문동 연애 시뮬레이션



15기 고태영, 김제성, 김지호
김지후, 정영희, 조우영

CONTENTS

프로젝트 소개

"아직까지 모솔인 나...내 남자친구는 어디있을까?! 후..나는 대화가 잘 통하는 사람을 만나고 싶은데...
갑자기 나타난 쌍문동 4인방!! 과연 내 사랑이 되어줄 사람은?!"

응답하라 1988 대본 데이터를 활용하여, **각 등장인물별 챗봇을** 제작한다.

드라마 상에서 등장인물의 특성을 드러낼 수 있는 챗봇을 구현하여

사용자와 채팅을 나눈 후, 연애 시뮬레이션 게임의 형태로 나에게 맞는 인물을 선정하는

것을 프로젝트 목표로 한다.

데이터 수집 및 전처리

1. 사용 데이터

응답하라 1988 대본 데이터

2. QA 형식의 데이터 구축

씬넘버, 배경, 등장인물 설명과 같은 필요없는 정보 삭제
질문(Q), 대답(A), 역할(role) 형식의 데이터 구축

A: 정환, 택, 선우, 동룡의 대사

Q: 그 앞의 대사

Role: A의 발화자

Q	A	role
야! 김정팔	꺼져.늦었어	정환

데이터 수집 및 전처리

3. 데이터 증강

2019년 EMNLP에서 발표된 **EDA**: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks 논문 참고

RS: Random Swap, 문장 내 임의의 두 단어의 위치를 바꿈
RD: Random Deletion: 임의의 단어를 삭제

(1) 문장의 순서 shuffle

(2) 단어 2개 shuffle

(3) 불필요한 단어 1개 삭제

(4) 코사인 유사도가 높은 문장

데이터 수집 및 전처리

3. 데이터 증강

Q

- original Q
- 문장 순서 shuffle
- 단어 2개 shuffle
- 불필요한 단어 1개 제거
- 코사인 유사도 top1
- 코사인 유사도 top2
- 코사인 유사도 top3

A

- original A
- 문장 순서 shuffle
- 단어 2개 shuffle
- 단어 1개 제거

야 김정팔!	꺼져. 늦었어.	정환
야 김정팔!	꺼져. 늦었어.	정환
야 김정팔!	늦었어. 꺼져.	정환
야, 김정환.	꺼져. 늦었어.	정환
야, 김정환.	꺼져. 늦었어.	정환
야, 김정환.	늦었어. 꺼져.	정환
야, 김정팔 들었지?	꺼져. 늦었어.	정환
야, 김정팔 들었지?	꺼져. 늦었어.	정환
야, 김정팔 들었지?	늦었어. 꺼져.	정환
어때?! 야, 김정팔!	꺼져. 늦었어.	정환
어때?! 야, 김정팔!	꺼져. 늦었어.	정환
어때?! 야, 김정팔!	늦었어. 꺼져.	정환
김정팔!! 야, 어때?!	꺼져. 늦었어.	정환
김정팔!! 야, 어때?!	꺼져. 늦었어.	정환
김정팔!! 야, 어때?!	늦었어. 꺼져.	정환
야, 김정팔!! 어때?	꺼져. 늦었어.	정환
야, 김정팔!! 어때?	꺼져. 늦었어.	정환
야, 김정팔!! 어때?	늦었어. 꺼져.	정환

Q에 7가지 데이터 증강 & A에 4가지 데이터 증강
하나의 QA 데이터에 대해 **28가지 버전의 데이터 생성됨**

유사도 기반 챗봇

1. 유사도 기반 챗봇

만약 사용자가 x라는 질문을 했다면, DB에서 x와 가장 비슷한 질문 Q를 찾아

그에 해당하는 대답 A를 출력하는 방식

텍스트의 유사도(cosine similarity)를 구하기 위해
Pororo를 이용해 Sentence embedding

```
sTe = Pororo(task="sentence_embedding", lang="ko")
data1['EmbVector'] = data1['Q'].progress_map(lambda x : sTe(x))
```

유사도 기반 챗봇

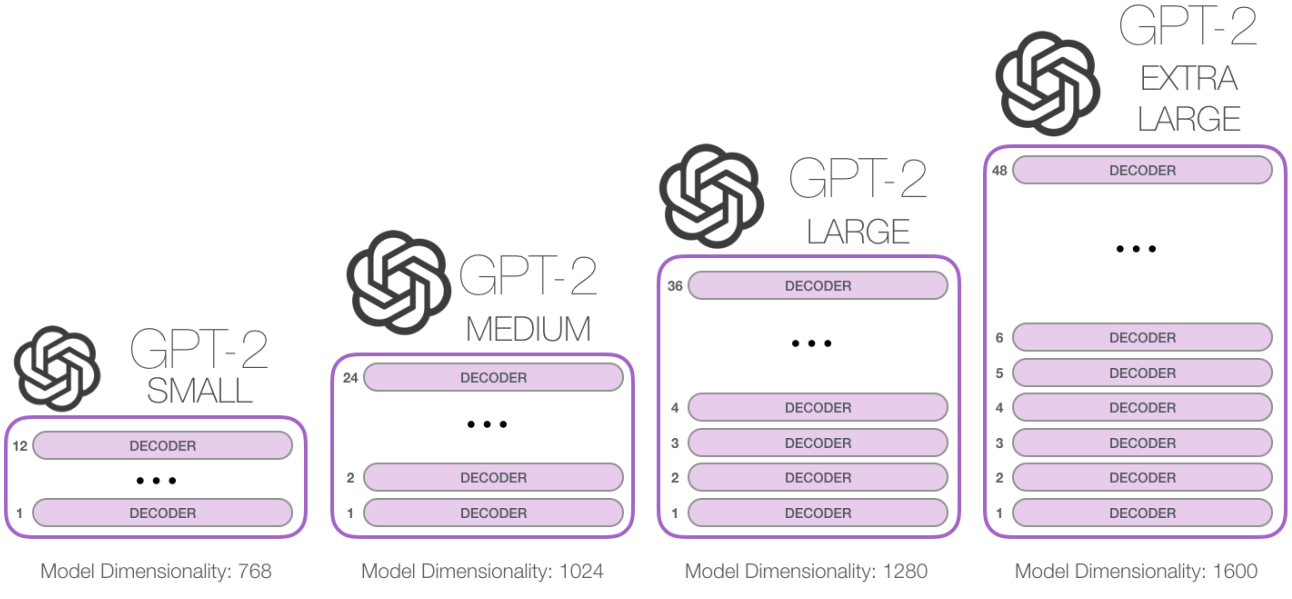
2. 코드

```
def chat():
  char = input("choose your character > ").strip()

  while 1:

    q = input("user > ").strip()
    if q == "quit":
      break
    a = ""
    # Pororo Sentense Embedding으로 텍스트 유사도를 구합니다.
    q = sTe(q)
    # 질문을 Tensor로 바꿉니다.
    q = torch.tensor(q)
    EmbData = torch.tensor(data1[data1['role']==char]['EmbVector'].tolist())
    # 코사인 유사도
    cos_sim = util.pytorch_cos_sim(q, EmbData)

    #유사도가 가장 비슷한 질문 인덱스를 구합니다.
    best_sim_idx = int(np.argmax(cos_sim))

    # 질문의 유사도와 가장 비슷한  답변 제공
    answer = data1['A'][best_sim_idx]
    print(answer)
```

KoGPT2 챗봇

1. GPT2

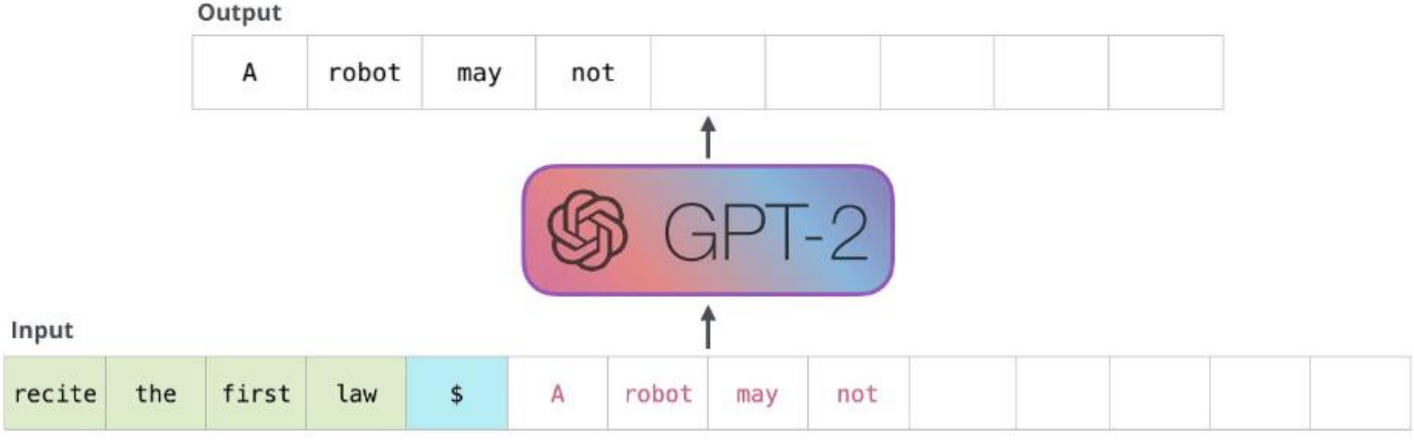
트랜스포머 구조에서 인코더를 제거하고 디코더로만 이루어진 모델



KoGPT2 챗봇

1. GPT2

각 토큰이 생성된 후, 입력 시퀀스에 더해지는 방식으로 동작하는 **auto-regression**
다음 단어의 예측 능력이 뛰어나 **생성 태스크에 적절**함.

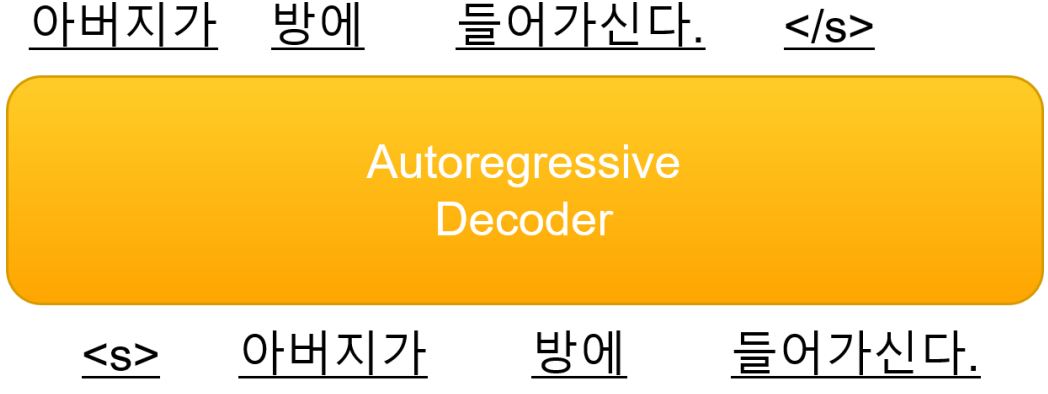


KoGPT2 챗봇

2. KoGPT2

KoGPT2는 부족한 한국어 성능을 극복하기 위해 40GB 이상의 텍스트로 학습된 한국어 디코더 언어모델.

"skt/kogpt2-base-v2" 의 사전학습 모델과 토크나이저를 사용



KoGPT2 챗봇

3. Fine-Tuning

기존에 학습되어 있는 모델을 기반으로 아키텍처를 새로운 목적에 맞게 변형하고 이미 학습된 모델 Weights로 부터 학습을 업데이트하는 방법

KoGPT2

한국어 위키 백과
뉴스, 모두의 말뭉치 v1.0
청와대 국민청원 등
다양한 데이터가 모델 학습에 사용됨

→

인물 성격 및 말투를
반영하기 위해
대본 데이터로
fine-tuning

KoGPT2 챗봇

4. 챗봇 데이터 클래스 정의

(1) Special token 정의

bos_token : 문장의 시작을 나타내는 token
eos_token : 문장의 끝을 나타내는 token
unk_token : 모르는 단어를 나타내는 token
pad_token : 입력의 크기를 동일하게 token
q_token : 질문의 시작을 나타내는 token
a_token : 답변의 시작을 나타내는 token

token_ids 순서는
+ 질문 + + 답변 + + pad_token_id

(2) 질문 및 답변 길이 조절

```
def __getitem__(self, idx):  # 로드한 챗봇 데이터를 차례차례 DataLoader로 넘겨주는 메서드
    turn = self._data.iloc[idx]
    q = turn["Q"]  # 질문 열을 가져온다.
    a = turn["A"]  # 답변 열을 가져온다.

    q_toked = self.tokenizer.tokenize(self.q_token + q  + self.bos)
    q_len = len(q_toked)

    a_toked = self.tokenizer.tokenize(self.a_token + a + self.eos)
    a_len = len(a_toked)

    #질문의 길이가 최대길이보다 크면
    if q_len > self.max_len:
        a_len = self.max_len - q_len        # 최대길이 - 질문길이
        if a_len <= 0:       # 질문의 길이가 너무 길어 질문만으로 최대 길이를 초과 한다면
            q_toked = q_toked[-(int(self.max_len / 2)) :]   # 질문길이를 최대길이의 반으로
            q_len = len(q_toked)
            a_len = self.max_len - q_len             #답변의 길이를 최대길이 - 질문길이
        a_toked = a_toked[:a_len]
        a_len = len(a_toked)
```

→ 최종적으로 token_ids, mask, labels_ids가 return됨

KoGPT2 챗봇

5. 코드

```
for epoch in tqdm(range(epoch)):
    for batch_idx, samples in enumerate(train_dataloader):
        optimizer.zero_grad()
        token_ids, mask, label = samples
        out = model(token_ids)
        out = out.logits
        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)
        # 평균 loss 만들기
        avg_loss = loss.sum() / mask.sum()
        avg_loss.backward()
        # 학습 끝
        optimizer.step()
```

최종 모델

1. 챗봇 결과

(2) 유사도 기반 챗봇

```
user > 지금 뭐해
어렸을 때는 술 한 모금도 못 마시더니
user > 너 술 좋아해?
아직 뉴스할 시간 안됐어. 경기 아직 끝나지도 않았을거고, 마감뉴스할 때 나오겠지.
user > 너 진짜 나쁜 아이구나
미쳤니? 택이 방.
user > 택이랑 놀고 싶어
너 평생 혼자 살 거냐?
user > 택이는 결혼하기 싫대?
아니. 아직도 안끝났대.
user > 걔도 정말 징하다
빼라고..!!
user > 선우는 지금 뭐하고 있어?
보자보자~
user > 너 진짜 말이 안 통하구나
오올~
user > 너 덕선이 좋아해>
어디가?
user > 덕선이 이쁘잖아
어디가?
user > 덕선이 싫어하는구나>
나? 아니지
user > 그럼 진짜 좋아해?
뭐?!
user > 정환이가 덕선이 좋아한대요~
어디가?
```

(2) KoGPT2 챗봇

```
user > 나는 덕선이야
Chatbot > 덕선이가 이쁘잖아. 더
user > 나도 예뻐
Chatbot > 하지마. 아무것도
user > 너무 추워
Chatbot > 아, 그러게 뭐 하러 입고 치마를 와..!!하여튼 애가 생각이 없어.
user > 좋아해
Chatbot > 옛날부터 얘기하고 싶었는데 나 진짜 좋아.
user > 너 미워
Chatbot > 그치!? 하긴 사람인데.너 너도 어떡하냐? 누나 남자친구 만나러 간다는데?
user > 아침 먹었어?
Chatbot > 어..!! 형이 제일 좋아하는 거..!!
user > 배고프다
Chatbot > 야. 택이 오랜만에 왔는데 라면이 먹자. 뭐야..시켜 어?
user > 학교가야지
Chatbot > 너랑 같이 학교가려고 매일 아침 대문 앞에서 1시간 넘게 기다리고
user > 나 기다렸어?
Chatbot > 오늘 저녁 뭐야, 형? 엄마 지금 뭐해?
user > 아 뭐래
Chatbot > 이거 아 팠어
user > 심심하다
Chatbot > 심심해..
user > 공부하자
Chatbot > 100원만  어머니,  주십쇼.
user > 놀러가자
Chatbot > 너 ...  놀래!?
```

최종 모델

2. 최종 모델

KoGPT2 모델 + 맞춤법 교정

- KoGPT2 모델

유사도기반 모델보다 대화가 자연스럽게 이어지는 결과 보임

- 맞춤법 교정

맞춤법 교정 패키지 Pororo를 사용해 답변 출력 이전 단계에서 교정
Spacing = Pororo(task="gec", lang ="ko")