REPORT

Final Report

학 번: 2017310367

이 름: 정다솔

제출일: 2022-6-10

1. 연구문제

동경하는 인물, 좋아하는 인물과 만나고 대화하고 싶은 것은 사람의 자연스러운 욕망이다. 실제로 동경하는 사람과 만나기 위해 사람들은 많은 돈을 지불하곤 한다. 유명 아이돌의 경우 팬사인회 한 번 5 분간의 대화를 위해 500 만원이 넘는 돈을 지불하기도 하고, 워런 버핏의 경우 한 끼의 저녁식사가 60 억에 팔렸다고 한다.

그렇지만 만약 만나고 싶은 인물이 이미 죽은 역사 속 사람이라면? 그림 속 모나리자라면? 신화 속 인물이라면? 혹은 셜록 홈즈와 같은 책 속의 등장인물이라면?

이와 같은 문제는 인문학을 좋아하는 사람에게 많이 발생하곤 한다.

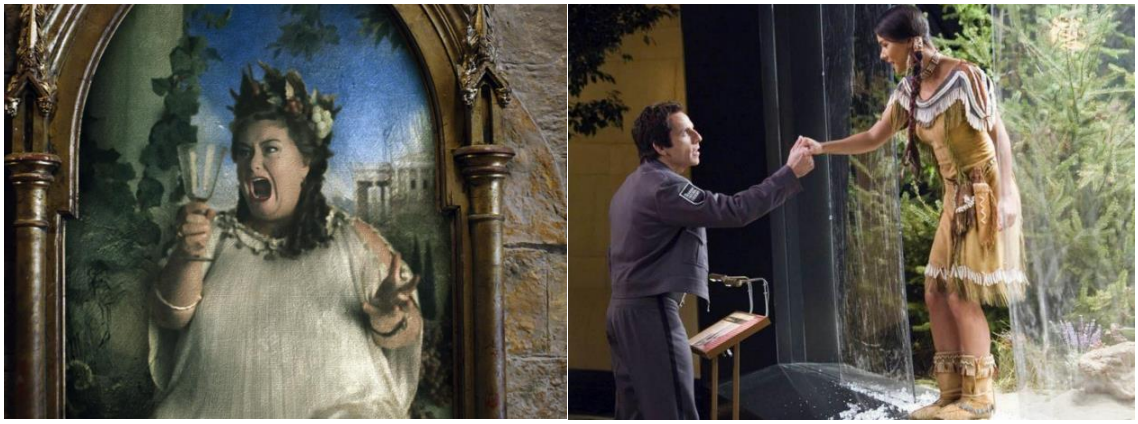
이 세상에 존재하지 않는 가상의 인물과 대화할 수는 없을까? 그게 책 속 등장인물인 셜록 홈즈라도?

과학 기술의 발전으로 인문학 또한 과학 기술의 발전에 힘입어 더 다양하게 구현되고, 연구되고, 발전하고 있다. 실제 박물관에서는 3D 프린터 모형을 통해 유물을 구현하여 시각장애인들의 관람에 도움을 주고 있으며, '안네의 일기'를 인공지능으로 분석하여 더 깊이 있는 분석에 도전하기도 한다. 이처럼 과학 기술을 통해서라면 인문학적 인물의 구현 또한 불가능한 것은 아니다.

이에 연구주제는 '실존하지 않는 사람(셜록 홈즈)을 인공지능을 통해 구현하기'이다.

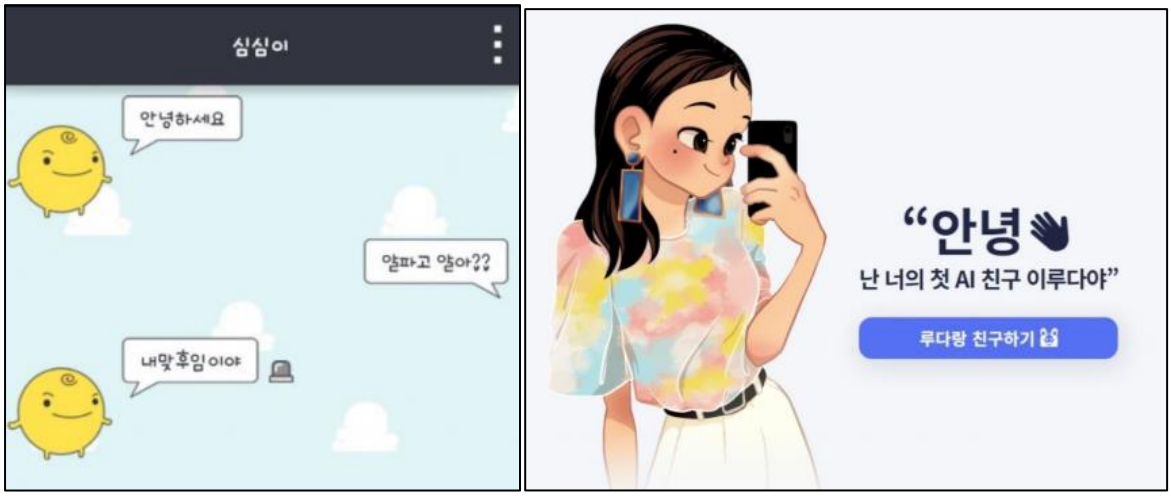
2. 선행연구

해당 연구주제와 가장 유사한 결과물은 판타지 소설와 영화에서 찾을 수 있었다.



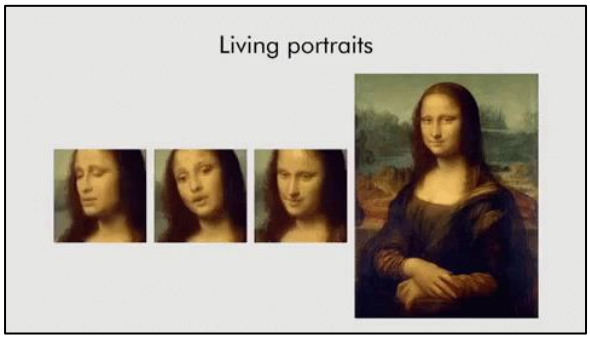
'해리포터 시리즈'에서는 이미 죽은 사람의 생각과 말투, 사상을 그대로 담은 초상화를 만들 수 있다는 설정이 존재한다. 이렇게 만들어진 초상화를 통해 등장인물들은 도움을 받거나 주기도 하며 더 이상 존재하지 않은 죽은 사람들과 상호작용한다. 또한 영화 '박물관이 살아있다'에서는 '사카주위아'에 대한 논문을 쓰고 있던 등장인물 '레베카'가 실제 생전의 모습과 생각을 가지고 있는 '사카주위아' 밀랍인형과 만나 논문에 대한 도움을 받는 내용이 나온다.

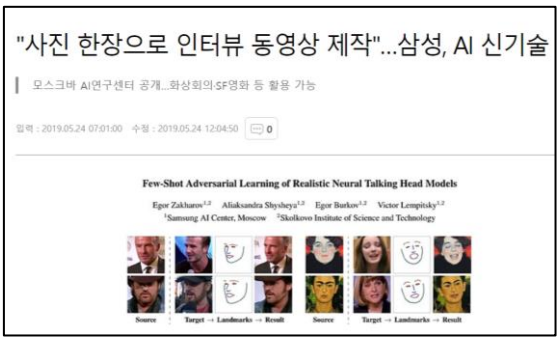
이처럼 존재하지 않는 사람과의 대화는 많은 사람들의 꿈이자 환상이었다. 그러나 아서 C. 클라크가 말했듯이, 충분히 발달한 과학 기술은 마법과 구별할 수 없다. 인공지능의 발달로 챗봇이 등장하면서 '존재하지 않는 무언가'와의 자연스러운 대화가 실제로 가능해졌다.



거대한 데이터셋을 바탕으로 한 '심심이'부터, 특정 자아를 가진 '이루다'까지 챗봇은 끊임없이 진화해왔다. 최근 '유니버스'에서는 특정 연예인의 말을 학습하여 그 연예인인 것처럼 대화를 할 수 있는 인공지능을 선보이기도 하였다.

이에 최근에는 SNS, 채팅 데이터를 기반으로 특정인 기반 AI 대화형 챗봇 기술이 나오기도 하였다.



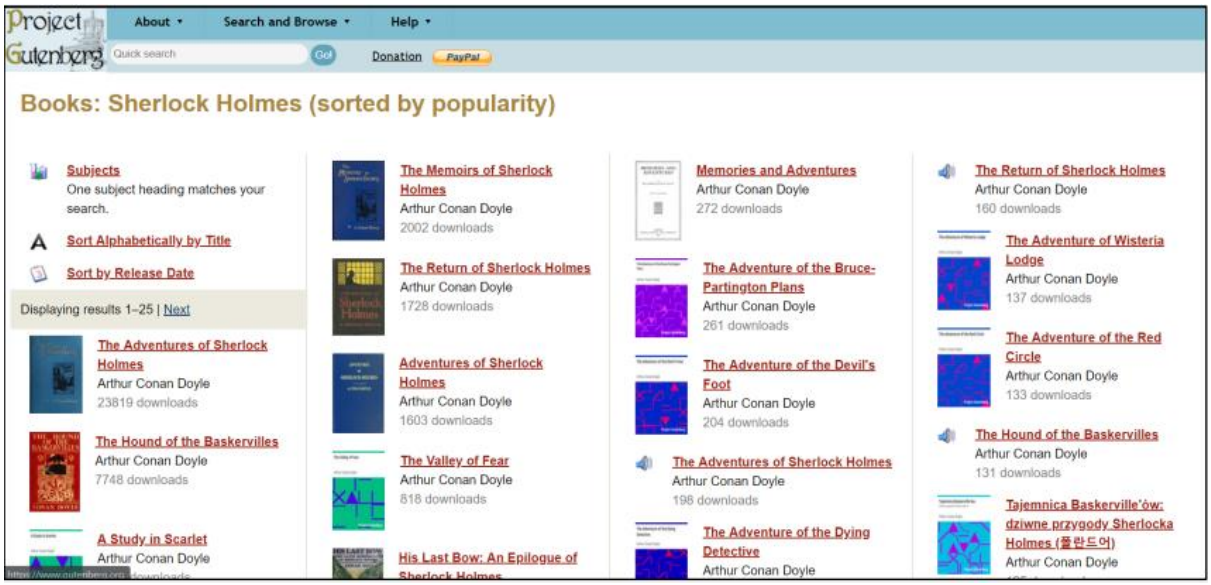


또한 삼성 연구소에서는 2019 년 사진 한장으로 동영상을 제작할 수 있는 인공지능 신기술 개발하였다. 실제 해당 기술을 바탕으로 모나리자가 자연스럽게 말하고 있는 것을 확인할 수 있다. 이처럼 딥페이크 기술 또한 점점 발전하여 최근에는 사진 한장만으로도 자연스러운 동영상을 만들 수 있는 수준이 되었다.

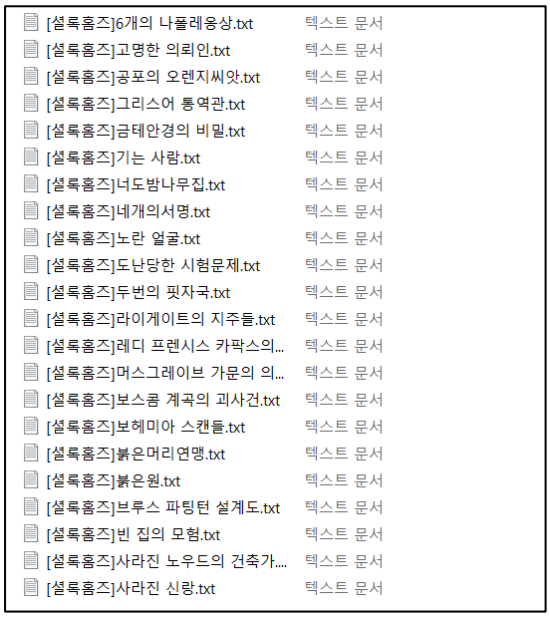
3. 수집 데이터

셜록 홈즈의 챗봇을 만들 계획이기 때문에 셜록 홈즈의 책 9 권을 수집할 데이터로 정한다. 셜록 홈즈의 책은 장편 4 권(주홍색 연구, 네 사람의 서명, 바스커빌 가의 개, 공포의 계곡), 단편 5 권(셜록 홈즈의 모험, 셜록 홈즈의 회상록, 셜록 홈즈의 귀환, 그의 마지막 인사, 셜록 홈즈의 사건집)으로 구성되어 있다.

소설 목록	
장편	주홍색 연구, 네 사람의 서명, 바스커빌 가의 개, 공포의 계곡
단편	
셜록 홈즈의 모험	보헤미아 스캔들 · 빨간 머리 연맹 · 신랑의 정체 · 보스콤 계곡 사건 · 다섯 개의 오렌지 씨앗 · 입술 삐뚤어진 사나이 · 푸른 카벙클 · 얼룩 띠의 비밀 · 어느 기술자의 엄지손가락 · 귀족 독신남 · 녹주석 보관 · 너도밤나무 집
셜록 홈즈의 회상록	실버 블레이즈 · 노란 얼굴 · 증권 거래소 직원 · 글로리아 스콧 호 · 머즈그레이브 가의 전례문 · 라이기트의 수수께끼 · 꼽추 사내 · 장기 입원 환자 · 그리스어 통역관 · 해군 조약문 · 마지막 사건
셜록 홈즈의 귀환	빈 집의 모험 · 노우드의 건축업자 · 춤추는 사람 인형 · 자전거 타는 사람 · 프라이어리 학교 · 블랙 피터 · 찰스 오거스터스 밀버턴 · 여섯 개의 나폴레옹 석고상 · 세 학생 · 금테 코안경 · 실종된 스리쿼터백 · 애비 그레인지 저택 · 두 번째 얼룩
그의 마지막 인사	등나무 집 · 소포 상자 · 붉은 원 · 브루스파팅턴 호 설계도 · 빈사의 탐정 · 프랜시스 카팍스 여사의 실종 · 악마의 발 · 마지막 인사
셜록 홈즈의 사건집	거물급 의뢰인 · 탈색된 병사 · 마자랭의 다이아몬드 · 세 박공 집 · 서섹스의 흡혈귀 · 세 명의 개리뎁 · 토르 교 사건 · 기어다니는 남자 · 사자의 갈기 · 베일 쓴 하숙인 · 쇼스콤 관 · 은퇴한 물감 제조업자



셜록 홈즈는 굉장히 인기가 많은 소설이고 저작권이 만료된 소설이라 쉽게 셜록 홈즈 데이터를 구할 수 있었다. 교수님이 알려주신 구텐베르크 홈페이지에서도 영어로 된 셜록 홈즈 원본을 전부 구할 수 있다.



또한 인터넷에서도 텍스트로 된 번역된 셜록 홈즈 전권을 확보할 수 있었다.

해당 파일은 모두 txt 파일로 이루어져있으며 9 권 모두 합쳐 2.20MB 의 용량으로 구성되어 있었다.

이렇게 얻은 텍스트 파일을 셜록 홈즈를 직접 읽어가며 대화 데이터를 추출하였다. 생각과는 달리 일상적인 대화보다는 추리에 관련된 대화가 많았고, 이 때문에 쓸 수 있는 대화쌍이 굉장히 한정되었다. '그 스폴딩이란 점원은 언제쯤부터 당신 가게에서 일하기 시작했습니까? 그리고, 어떤 인연으로 알게 되었습니까?' 이런 추리 관련한 대화들은 사용할 수 없었으나 좋아하는 음악을 설명하는 부분에서의 '쇼팽 소품 연주가 굉장합니다. 트라~라~라--리라~ 레이.'와 같은 대화는 사용할 수 있었다.

실제 데이터의 예시는 다음과 같다.

데이터 예시

존 왓슨: 그렇다면 자네는 이 방에서 한 발자국도 나가지 않고도 사건에 휘말려 있는 본인도 풀 수 없는 의문을 풀어 줄 수가 있단 말인가?

셜록 홈즈: 물론이지. 그런 일에 관해선 나는 일종의 직감을 갖고 있다네. 그야 때에 따라서는 복잡한 사건도 있기 마련이지.

레스트레이드 경감: 사건은 더욱 얽히고 설키는군. 그러잖아도 복잡했는데...

셜록 홈즈: 그 덕분에 오히려 사건이 간단해질지도 모르는 일입니다.

- 문맥을 위한 셜록 홈즈 책의 전체 데이터 사용
- 문답을 위해 셜록 홈즈와의 대화 데이터 추출 후 사용
- 셜록 홈즈 성격을 반영하기 위한 셜록 홈즈의 묘사, 대사 데이터 사용

그 결과 셜록 홈즈 책에서 약 1200 개의 대화쌍을 추출할 수 있었다. 그러나 챗봇 인공지능을 학습시켜본 결과 해당 데이터셋은 심각하게 적었다. 이에 인공지능이 셜록 홈즈의 자아를 보여주기는커녕 챗봇으로서 제대로 작동하지도 않는 모습으로 보였다.

ChatbotData

Q	A	label
12시 땡!	하루가 또 가네요.	0
1지망 학교 떨어졌어	위로해 드립니다.	0
3박4일 놀러가고 싶다	여행은 언제나 좋죠.	0
3박4일 정도 놀러가고 싶다	여행은 언제나 좋죠.	0
PPL 심하네	눈살이 찌푸려지죠.	0
SD카드 망가졌어	다시 새로 사는 게 마음 편해요.	0
SD카드 안돼	다시 새로 사는 게 마음 편해요.	0

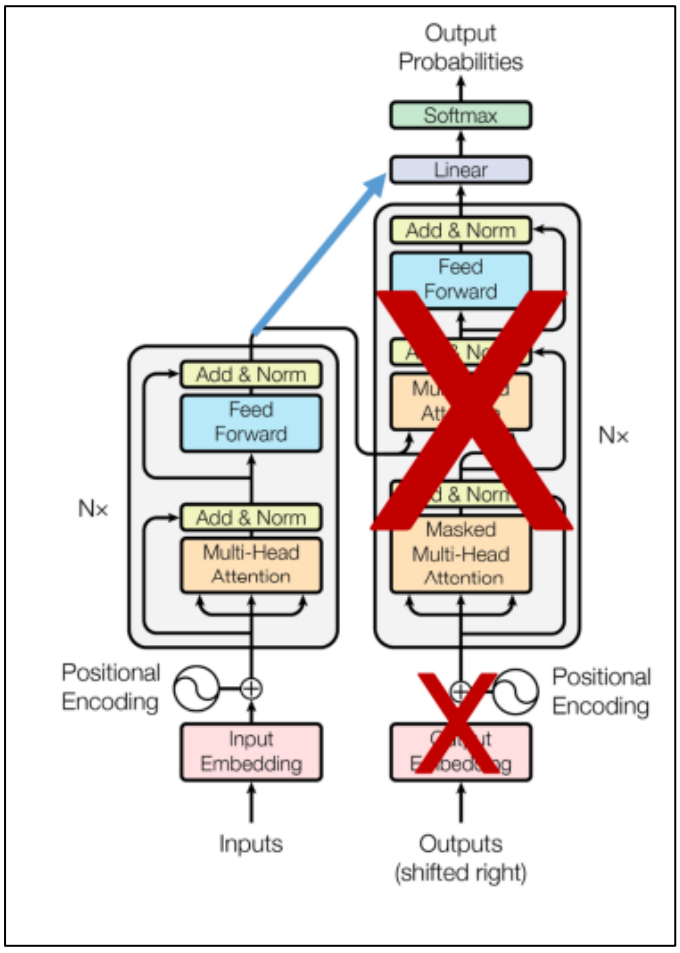
이에 챗봇으로서 자연스러운 대화가 가능하도록 일반 챗봇데이터를 추가적으로 사용하였다. 추가된 데이터는 챗봇 트레이닝용 문답 페어 11,876 개를 가지고 있는 챗봇 데이터로 질문인 Q 필드와, 질문에 대한 대답인 A 필드, 그리고 감정 레이블로 구성되어 있다. 질문에 대한 대답의 경우 말투와 성격이 셜록 홈즈와 유사하도록 모두 바꿔주었다.

4. 분석방법

챗봇을 만들기 위한 인공지능으로는 GPT(Generative Pre-trained Transformer)와 BERT(Bidirectional Encoder Representations from Transformers)를 고려하였다. NLP 에서는 seq2seq 의 문제점을 해결하기 위해 Attention 을 사용한다. Attention 은 RNN 의 vanishing gradient 문제 와 context vector 의 정보 손실 문제를 해결하기 위해 사용되고 있다. 이러한 Attention 의 응용이 바로 Transformer 로 GPT 와 BERT 는 모두 Transformer 를 사용한다.

GPT 는 이전 이전 단어들이 주어졌을 때 다음 단어가 무엇인지 맞추는 과정에서 학습을 진행하는 모델로 문장 시작부터 순차적으로 계산한다는 점에서 일방향의 성격을 가지고 있다. 반면 BERT 는 문장 중간에 빈칸을 만들고 해당 빈칸에 어떤 단어가 적절할지 맞추는 과정에서 학습을 진행하는 모델로 빈칸 앞뒤 문맥을 모두 살필 수 있다는 점에서 양방향의 성격을 가진다. 또한 GPT 는 트랜스포머에서 디코더만을, BERT 는 트랜스포머에서 인코더만을 사용한다는 구조적인 차이가 있다.

BERT 의 구조는 다음과 같다.



입력 단어 시퀀스가 '어제 도서관 갔어. 이 책은 [A] 빌려왔어.'라고 가정한다면, BERT 는 A 토큰 앞뒤 문맥을 모두 참고한다.

A 라는 단어에 대응하는 BERT 마지막 레이어의 출력 결과에 계산을 수행하여 어휘 수만큼의 확률 벡터를 만든다. 그 후 A 의 정답인 '도서관에서'에 해당하는 확률은 높이고 나머지 단어의 확률은 낮아지도록, 모델 전체를 업데이트한다. 이러한 방법으로 학습을 진행한다.

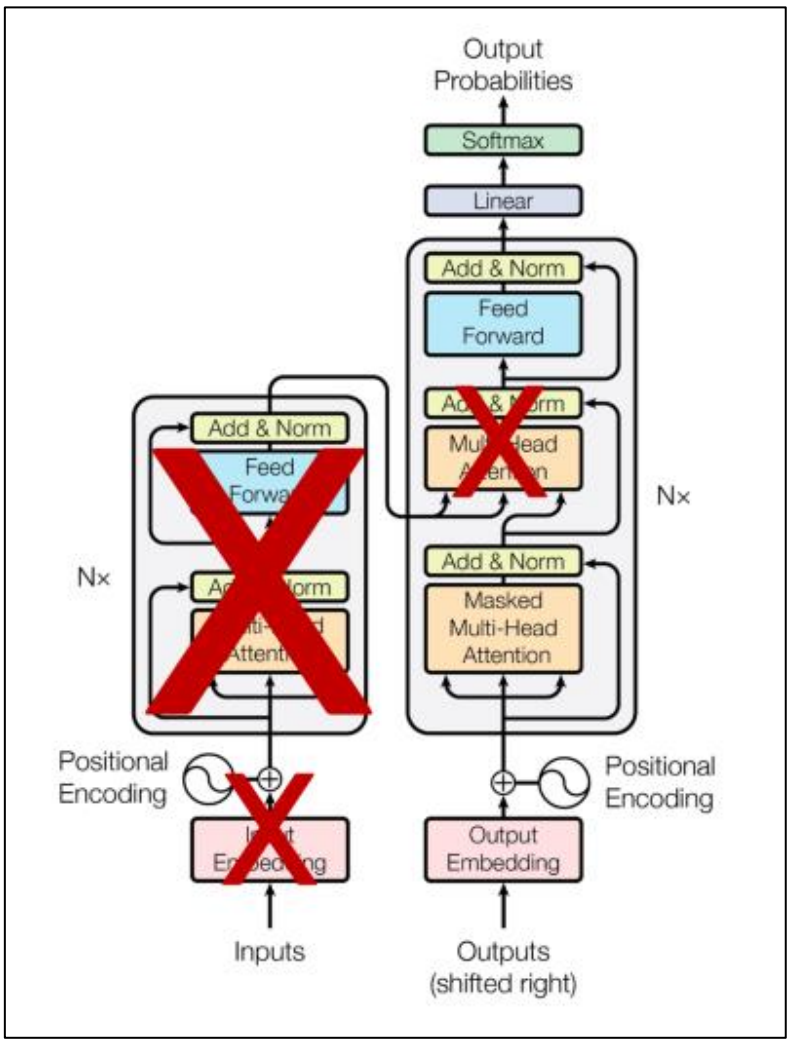
이 때문에 BERT 는 문장의 의미를 추출하는 데에 강점을 지닌 것으로 알려져 있다. 그러나 해당 모델은

사용하지 않았다. 그 이유는 다음과 같다.

1. BERT 는 문맥파악에, GPT 는 문장 생성에 강점을 가지고 있기 때문에 챗봇 제작에는 BERT 보다는 GPT 가 유리하다
2. 어차피 해당 프로젝트의 문제는 데이터의 부족이라 BERT 로도 해결이 불가능하다.

이에 챗봇 모델 자체는 GPT 로 결론을 내린 후 챗봇의 성능을 높이기 위해 챗봇데이터를 추가함으로써 데이터의 부족을 해결하는 방안을 찾아냈다.

GPT 의 구조는 다음과 같다.



GPT 는 트랜스포머에서 인코더를 제외하고 디코더만 사용한다.

디코더 블록을 자세히 보면 인코더 쪽에서 보내오는 정보를 받는 모듈 역시 제거돼 있음을 확인할 수 있다.

입력 단어 시퀀스가 '어제 [A] 갔어. 이 책은 도서관에서 빌려왔어.'라고 가정한다면, GPT 는 정답인 단어 '도서관'을 맞추기 위해 어제라는 단어만 참고할 수 있다.

따라서 정답 단어 이후의 모든 단어(갔어~ 빌려왔다)는 참고할 수 없으며 볼 수 없도록 해야 한다. 이에 해당 단어들에는 Score 가 0 이 되도록 하는 값인 Mask 를 씌운다.

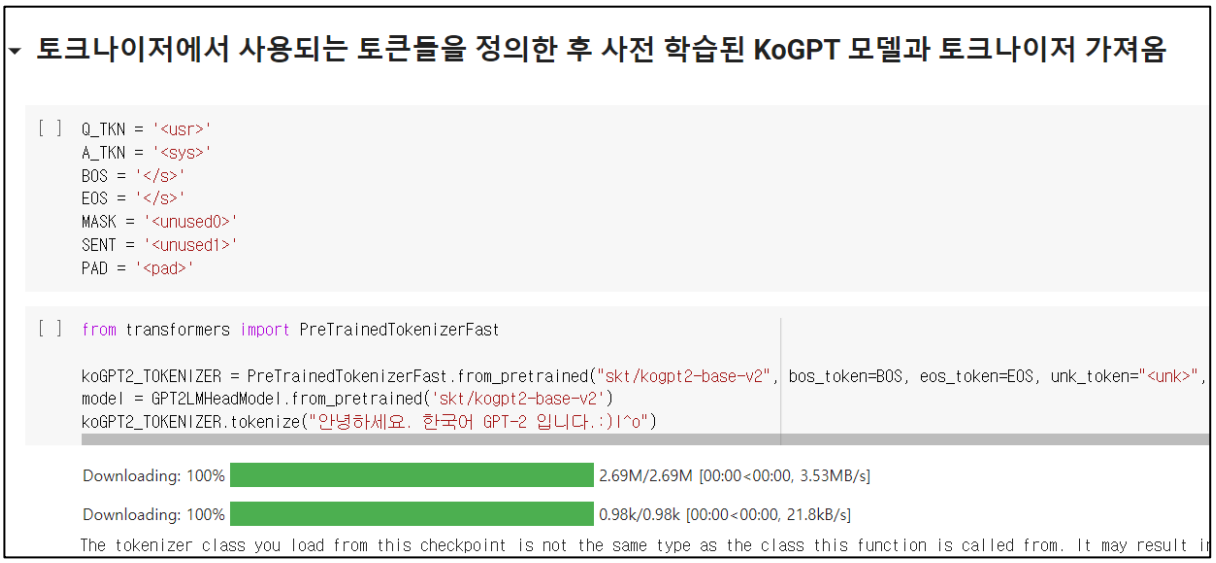
어제라는 단어에 대응하는 GPT 마지막 레이어의 출력 결과에 계산을 수행하여 어휘 수만큼의 확률 벡터를 만든다. 그 후 A 의 정답인 '도서관'에 해당하는 확률은 높이고 나머지 단어의 확률은 낮아지도록, 모델 전체를 업데이트한다.

그 후 '갔었어'라는 단어를 맞추기 위해 GPT 는 '어제'와 '도서관'이라는 단어를 참고할 수 있다. 정답인 '갔었어'에 해당하는 확률은 높이고 나머지 단어의 확률은 낮아지도록, 모델 전체를 업데이트한다. 이러한 방법으로 학습을 진행한다.

실제 이번 프로젝트에서 구현한 챗봇은 GPT-2 를 파인튜닝한 KoGPT 를 사용하였다.

2020 년 SKT-AI 에서 GPT-2 모델을 Fine-Tuning 한 한국어 언어모델로 KoGPT 를 제공하고 있다. KoGPT 는 한국어 위키 백과, 청와대 국민청원 등을 포함한 데이터 셋을 학습에 사용하였으며, ChatBot 예제 코드를 제공하기 때문에 이번 프로젝트를 구현하는 데에 용이하다는 장점이 있다.

실제 구현 코드 설명은 다음과 같다.



가장 먼저, skt 에서 제공하는 KoGPT 모델을 가져온다. 해당 모델에서 사용하는 토크나이저를 그대로 사용할 예정이기 때문에 토크나이저에서 사용되는 토큰들을 사전에 정의해주었다.

```
    if q_len + a_len > self.max_len:
        a_len = self.max_len - q_len
        if a_len <= 0:
            q_toked = q_toked[-(int(self.max_len / 2)) :]
            q_len = len(q_toked)
            a_len = self.max_len - q_len
        a_toked = a_toked[:a_len]
        a_len = len(a_toked)

    labels = [self.mask,] * q_len + a_toked[1:]

    mask = [0] * q_len + [1] * a_len + [0] * (self.max_len - q_len - a_len)
    labels_ids = self.tokenizer.convert_tokens_to_ids(labels)
    while len(labels_ids) < self.max_len:
        labels_ids += [self.tokenizer.pad_token_id]

    token_ids = self.tokenizer.convert_tokens_to_ids(q_toked + a_toked)

    while len(token_ids) < self.max_len:
        token_ids += [self.tokenizer.pad_token_id]

    return (token_ids, np.array(mask), labels_ids)
```

그 후 데이터로더를 만든다. 데이터로더는 셜록 홈즈 챗봇 학습에 사용될 글자 데이터들을 인공지능 학습용 벡터로 바꿔주는 역할을 한다. 자연어는 모델의 Input 이 될 수 없으며 벡터값으로 바꾸는 과정이 필요하다. 그 첫 단계가 Tokenize 으로, 각 단어에게 구별되는 고유한 Index 를 붙여주는 작업을 데이터로더에서 진행한다. 모든 단어들에 다른 Index 를 부여할 수는 없으며 모델이 처리할 수 있는 단어에만 Index 를 정해주는데, 이를 Vocab 이라고 한다. 데이터에 Vocab 에 존재하지 않는 단어가 나오게 되면, 이를 [UNK] 토큰으로 처리한다. GPT 에서는 특히 한 단어를 여러 토큰으로 분리하는 Subword Tokenize 방식을 사용한다. 또한 데이터로더에서는 기준에 맞게 질문 길이를 조절하였으며, 질문 및 답변 길이가 기준에 비해 짧을 경우 패딩을 삽입하였다. 또한 느낌표, 물음표, 쉼표, 마침표와 같은 구두점을 제거하였다.

```
        labels = [self.mask,] * q_len + a_toked[1:]

        mask = [0] * q_len + [1] * a_len + [0] * (self.max_len - q_len - a_len)
        labels_ids = self.tokenizer.convert_tokens_to_ids(labels)
        while len(labels_ids) < self.max_len:
            labels_ids += [self.tokenizer.pad_token_id]

        token_ids = self.tokenizer.convert_tokens_to_ids(q_toked + a_toked)

        while len(token_ids) < self.max_len:
            token_ids += [self.tokenizer.pad_token_id]

        return (token_ids, np.array(mask), labels_ids)
```

그 결과로 나오게 되는 return 값은 다음과 같다.

- Token_id : 질문 + 감정 + 답변
- Mask : mask
- Label_id : 답변

Token id 는 모델 학습에 들어가는 input 값이다. Mask 는 특정 토큰을 Attention 에서 가리기 위해서 사용된다. GPT 에서는 다음에 올 토큰을 예측할 때 이전의 토큰들과의 Attention 만을 사용해서 예측하기 때문에, 뒤에 오는 토큰들의 Attention 을 가리는 Mask 를 필요로 한다. Label id 는 모델 학습 output 과 비교하여 모델을 학습하는 데에 사용한다.

모델 학습

```
[ ]  model.to(device)
     model.train()
```

```
[ ]  learning_rate = 3e-5
     criterion = torch.nn.CrossEntropyLoss(reduction="none")
     optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

     epoch = 10
     Sneg = -1e18
```

모델 학습을 시작한다. KoGPT 모델을 셜록 홈즈 챗봇으로 Fine-tunning 하는 작업으로 손실함수는 Cross entropy, 옵티마이저는 Adam 을 사용하였다. Epoch 수는 10 다.

```
print ("start")
for i in range(0, epoch):
    for batch_idx, samples in enumerate(train_dataload
        optimizer.zero_grad()
        token_ids, mask, label = samples
        token_ids = token_ids.to(device)
        out = model(token_ids)
        out = out.logits
        mask_3d = mask.unsqueeze(dim=2).repeat_interle
        mask_out = torch.where(mask_3d == 1, out.cpu()
        loss = criterion(mask_out.transpose(2, 1), lab
        avg_loss = loss.sum() / mask.sum()
        avg_loss.backward()
        optimizer.step()
print ("end")
```

GPT 는 input 으로 들어간 token id 로 Query, Key, Value 를 만든다. 이때 Query, Key, Value 는 hidden_size 에서 hidden_size 로의 Linear Layer 을 거쳐 만들어지며, 실제 GPT 가 학습하는 것은 바로 이 Query, Key, Value 를 만드는 Layer 이다. 그 후 각 토큰의 Query, Key, Value 를 여러 개의 작은 Query, Key, Value 로 분할된다. 분할된 토큰의 Query 는 해당 Sample 의 모든 Key 들과 Matrix Multiplication 을 하게 되는데, 이 결과를 Score 라고 한다.

그 후 Mask 를 이용하여 Attention 에 마스킹을 한다. GPT 는 자기 이후의 토큰에 대한 Attention 값을 사용하지 않는다. 따라서 위 Score 은 각 토큰이 모든 다른 토큰들에 대한 Attention 값을 가지고 있기 때문에 수정할 필요가 있다. 큰 음수를 곱한 후 Attention Mask 를 Score 에 더한 뒤 Softmax 를 취하게 되면 Mask 에서 True 였던 부분이 0 이 되며, 자기 이후의 토큰들에 대한 Attention 값을 삭제하는 효과가 있다.

이렇게 나온 Output 벡터를 사용하여 다음 토큰을 예측하기 위해서는 모델에 있는 각 토큰이 정답이 될 확률을 구한다. Output Embedding 을 거쳐서 나온 Logit 에 Softmax 를 취해 다음에 각 토큰이 올 확률을 구하게 된다. 이후 해당 값과 Label Id 의 CrossEntropy 를 사용하여 loss 를 구한다.

챗봇 실행

```
sent='0'
a = ""
sent_tokens = koGPT2_TOKENIZER.tokenize(sent)
tmp = 0
with torch.no_grad():
    while 1:
        q = input("user > ").strip()
        if q == "quit":
            break
        if q == "save":
            tmp = tmp + 1
            speak(a.strip(), tmp)
            continue
```

```
user > 너한테만 쉬운 거 아닐까?
Sherlock > 저도 어려운 게 투성입니다
user > 어쨌든 응원해줘서 고마워
Sherlock > 응원합니다
user > 얼른 시험 끝나고 야구보러 가고싶다
Sherlock > 좋은 결과 있을 겁니다
user > 이번엔 우리팀이 이겼으면 좋겠어
Sherlock > 끝까지 잘 해보십시오
user > 나말고 야구선수들이 잘해야지
Sherlock > 약점 분석해보십시오
user > 어쨌든 이번학기 화이팅이야!
Sherlock > 생각한대로 잘할 수 있을 겁니다
```

이렇게 만들어진 챗봇을 실행한 결과 어색한 부분이 있긴 하지만 기본적인 대화가 가능한 것으로 보였다.

그러나 해당 챗봇은 몇 가지 한계를 가지고 있다.

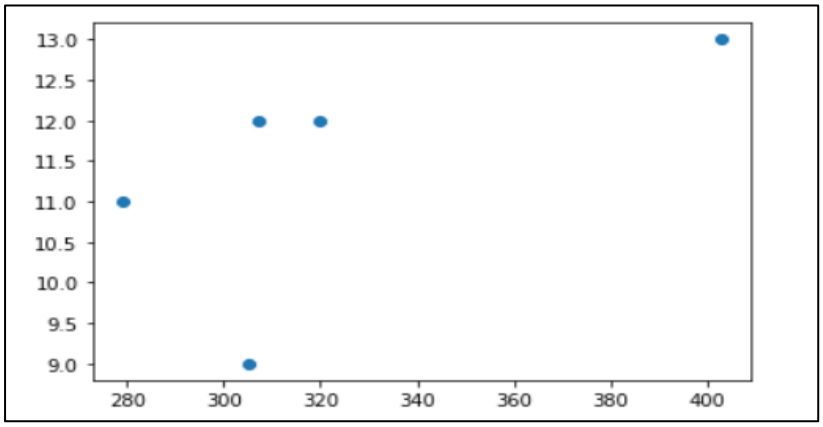
1. 대화의 맥락을 이어서 말하지 못한다. 야구경기를 보러가는데 '우리팀'을 '내가 응원하는 야구팀'으로 해석하지 못했다.
2. 학습시킬 때 물음표와 느낌표, 쉼표와 같은 구두점들을 제거했더니 '오늘 뭐 먹었어.'와 '오늘 뭐 먹었어?'를 구분하지 못한다. 이에 질문을 해도 제대로 된 대답이 나오지 않은 것으로 보인다.
3. 셜록 홈즈의 챗봇이니만큼 수준 높은 추리실력을 보여주었으면 하는데 아직 맥락을 잡는 것부터가 부족하다. 셜록 홈즈가 되려면 기술이 한참 부족하다.

Linear regression

셜록 홈즈 책의 페이지와 에피소드의 개수를 Data Frame 으로 만들었다. 셜록 홈즈의 단편은 총 5 권으로 각 책의 페이지 수와 에피소드의 개수는 다음과 같다.

	book	episode	page
0	The Adventures of Sherlock Holmes	12	307
1	The Memoirs of Sherlock Holmes	11	279
2	The Return of Sherlock Holmes	13	403
3	His Last Bow	9	305
4	The Case-Book of Sherlock Holmes	12	320

해당 데이터의 산포도는 다음과 같다.



에피소드의 개수가 많으면 페이지의 수 또한 많을 것이라는 가설을 세운 후 Linear regression 을 진행하였다. 그 결과는 다음과 같다.

OLS Regression Results

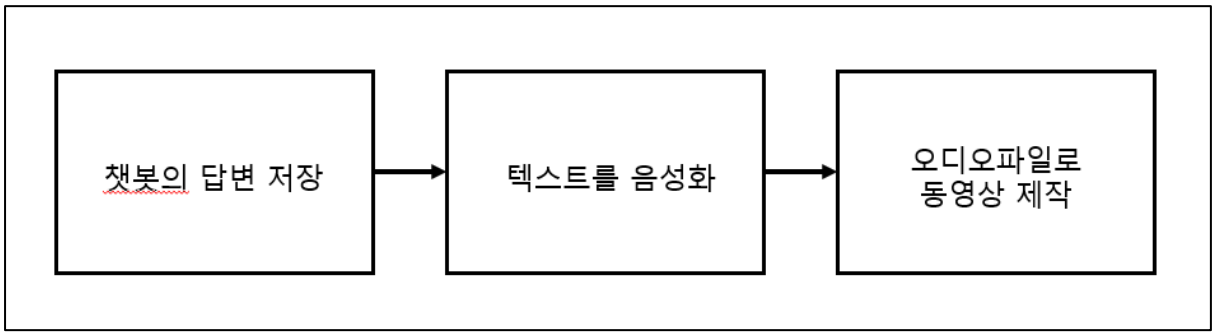
Dep. Variable:	episode	**R-squared:**	0.383
Model:	OLS	**Adj. R-squared:**	0.178
Method:	Least Squares	**F-statistic:**	1.864
Date:	Thu, 02 Jun 2022	**Prob (F-statistic):**	0.265
Time:	19:09:00	**Log-Likelihood:**	-7.4107
No. Observations:	5	**AIC:**	18.82
Df Residuals:	3	**BIC:**	18.04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.9836	4.739	1.052	0.370	-10.098	20.066
page	0.0199	0.015	1.365	0.265	-0.026	0.066

Omnibus:	nan	**Durbin-Watson:**	2.101
Prob(Omnibus):	nan	**Jarque-Bera (JB):**	1.259
Skew:	-1.227	**Prob(JB):**	0.533
Kurtosis:	2.867	**Cond. No.**	2.51e+03

5. 시각화

'셜록 홈즈가 살아 움직이는 것을 만드는 것'이 목표인 이상 시각화는 셜록 홈즈가 실제로 말하는 동영상을 만들기로 하였다.



시각화의 목표는 다음과 같다. 가장 먼저, 해당 챗봇에서 'save'를 입력하면 해당 답변은 저장이 된다. 이렇게 저장된 답변 텍스트를 Text-to-speech 기술을 사용하여 음성화하고, 이렇게 만들어진 오디오파일을 사용하여 셜록 홈즈가 말하는 동영상을 제작한다.

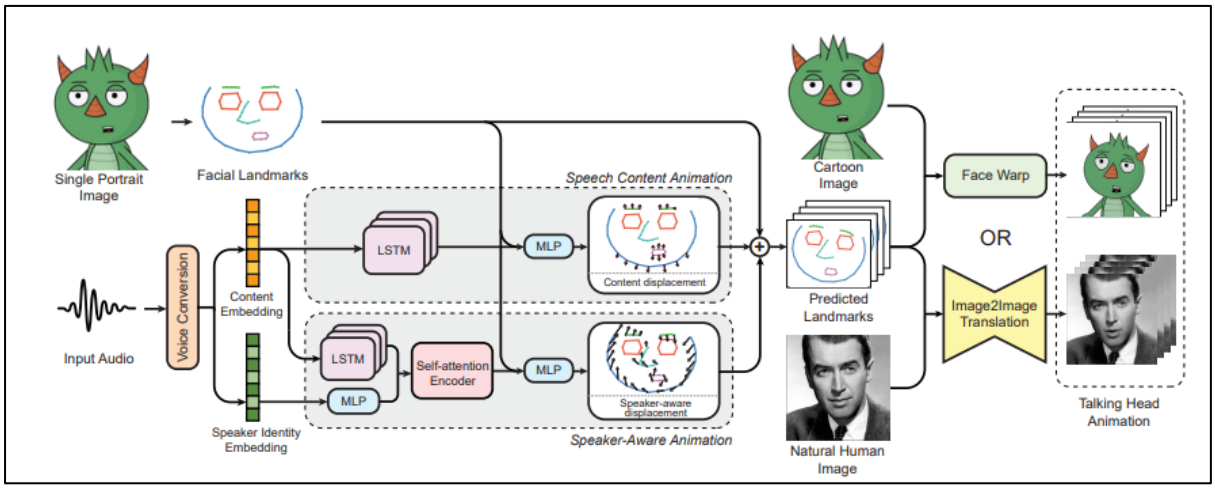
실제 구현은 다음과 같다.

텍스트를 음성파일로 변환

```
[37]  import speech_recognition as sr
      from gtts import gTTS
      import os
      import time

      def speak(text, tmp):
          tts = gTTS(text=text, lang='ko', slow=False)
          filename="/content/MakeItTalk/examples/"+str(tmp)+'.wav'
          tts.save(filename)
```

구글 텍스트 음성 변환 모듈인 gTTS 를 사용하여 텍스트를 음성파일로 변환하였다.



그 후 만들어진 음성파일을 셜록 홈즈의 얼굴이 말하도록 하기 위해, 2021 년 메사추사스 대학과 어도비에서 발표한 'MakeItTalk: Speaker-Aware Talking-Head Animation' 논문에 기반한 딥페이크 기술을 사용하였다. 논문에 의하면 해당 기술은 음성파일을 MLP 와 LSTM 을 사용하여 분석한 후 랜드마크 추출하고, 그 후 Image2Image Translation 을 사용하여 사진과 결합하는 방식을 사용한다.

셜록 홈즈의 사진의 경우, 팬들에게 서 가장 셜록 홈즈다운 셜록 홈즈를 연기했다는 평가를 받는 그라나다 TV 의 셜록 홈즈 드라마 제레미 브렛의 작중 이미지를 사용하였다.

```
fl_data = []
rot_tran, rot_quat, anchor_t_shape = [], [], []
for au, info in au_data:
    au_length = au.shape[0]
    fl = np.zeros(shape=(au_length, 68 * 3))
    fl_data.append((fl, info))
    rot_tran.append(np.zeros(shape=(au_length, 3, 4)))
    rot_quat.append(np.zeros(shape=(au_length, 4)))
    anchor_t_shape.append(np.zeros(shape=(au_length, 68 * 3)))
```

실제 구현된 코드에서 사진의 랜드마크를 잡아주는 것을 확인할 수 있다.

```
make_model = Image_translation_block(opt_parser, single_test=True)
with torch.no_grad():
    make_model.single_test(jpg=img, fls=fl, filename=fls[i], prefix=opt_parser.jpg.split('.')[0])
    print('finish image2image gen')
os.remove(os.path.join('examples', fls[i]))
```

Image2Image Translation 을 사용하여 사진과 결합하는 것을 확인할 수 있다.

그 결과 셜록 홈즈가 말하는 동영상을 제작할 수 있었다.

6. 조원 역할 분담

역할 분담은 다음과 같다.

역할	담당자
데이터 수집 (셜록 홈즈 책)	정다솔
데이터 정제 (셜록 홈즈 대사 추출)	정다솔
데이터 분석 (챗봇 인공지능 제작)	정다솔
데이터 시각화 (초상화 인공지능 제작)	정다솔
발표 및 보고서 작성	정다솔

7. 수업에 대한 피드백

- 수업에서 좋았던 점

  (좋은 점 1 가지 너무 적다고 생각합니다...) 교수님이 주신 과제가 굉장히 좋았습니다. 살짝 어려우면서도 그날 배운 내용을 응용하는 과제들이 많아서 과제를 하는 동안 많이 배운 것 같습니다. 교수님의 설명도 쉽고 재미있어서 좋았습니다. 네트워크나 확률같이 평소에는 접하지 못한 새로운 분야들 또한 접할 수 있어서 좋았습니다.

- 수업에서 개선해야 할 점

  (3 가지는 너무 많습니다...)

  1. 수업 난이도가 쉬운 부분은 쉽고 어려운 부분은 어려웠습니다. 제가 잘 모르는 분야는 어렵게 느껴지고 잘 아는 분야는 쉽게 느껴졌습니다.

  2. 저는 한 가지 분야를 조금 더 깊게 배우고 싶었는데 그 부분은 아쉬웠습니다. 특히 네트워크와 인공지능 분야가 흥미로웠습니다.

  3. 초반 파이썬 부분이 빠지고 해당 내용이 늘면 좋겠지만 그건 안되겠죠....

8. Reference

KoGPT2 코드

- KoGPT2 (한국어 GPT-2) Ver 2.0, https://github.com/SKT-AI/KoGPT2
- Simple Chit-Chat based on KoGPT2, https://github.com/haven-jeon/KoGPT2-chatbot
- 9-3. koGPT2 ChatBot, https://wikidocs.net/158023

챗봇데이터

- Chatbot_data., https://github.com/songys/Chatbot_data

동영상제작

- MakeItTalk: Speaker-Aware Talking-Head Animation, https://github.com/yzhou359/MakeItTalk

GPT 와 BERT 관련

- GPT (Generative Pre-trained Transformer) 학습시키기, https://ainote.tistory.com/17
- BERT 와 GPT, https://ratsgo.github.io/nlpbook/docs/language_model/bert_gpt/