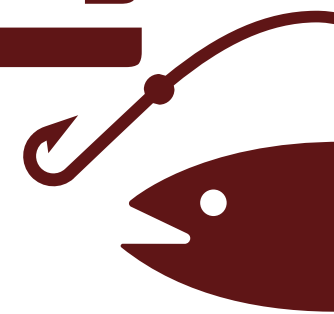


LLM을 활용한
보이스피싱 예방
웹 어플리케이션
프로젝트



박거량 박소진 백지영 이진영



목차



- 프로젝트 개요
- 아키텍처 소개
- 데이터 소개
- 모델링
- 의문점
- 배포 웹사이트 시연
- Q&A



프로젝트 개요



LLM을 활용한 보이스피싱 예방 웹 어플리케이션

1. 보이스피싱 판별

- 통화 음성 파일 입력 시 통화내용이 보이스피싱일 확률 제시

2. 보이스피싱 롤플레이잉

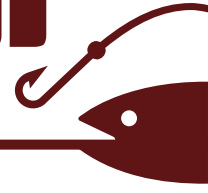
- 사용자가 선택한 보이스피싱 상황에 대해 롤플레이잉
- 대화 종료 시 사용자의 대응에 대한 피드백 제시

3. 보이스피싱 대처방안

- 여러 보이스피싱 피해 상황에 대한 대처방안 제시



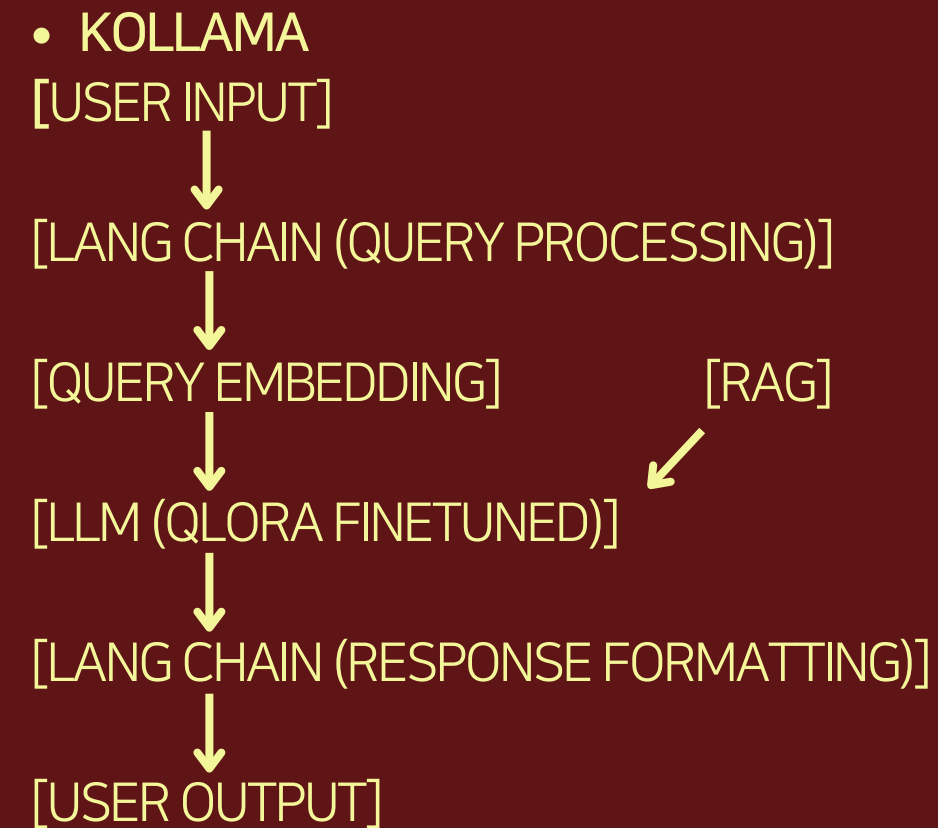
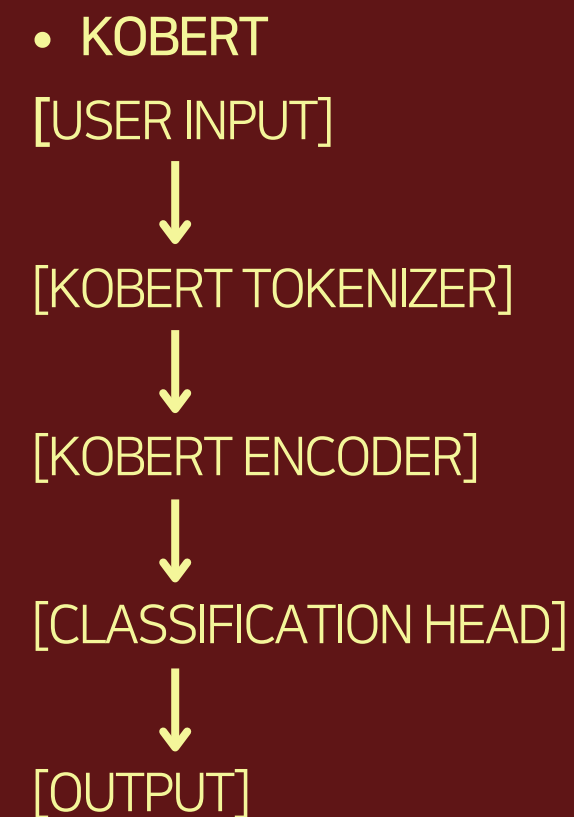
아키텍처 소개



FRONT-END

- REACT를 이용한 비동기적 통신
- 코드 모듈화 -> 코드의 유연성 확보
- 반응형 UI

MODELING

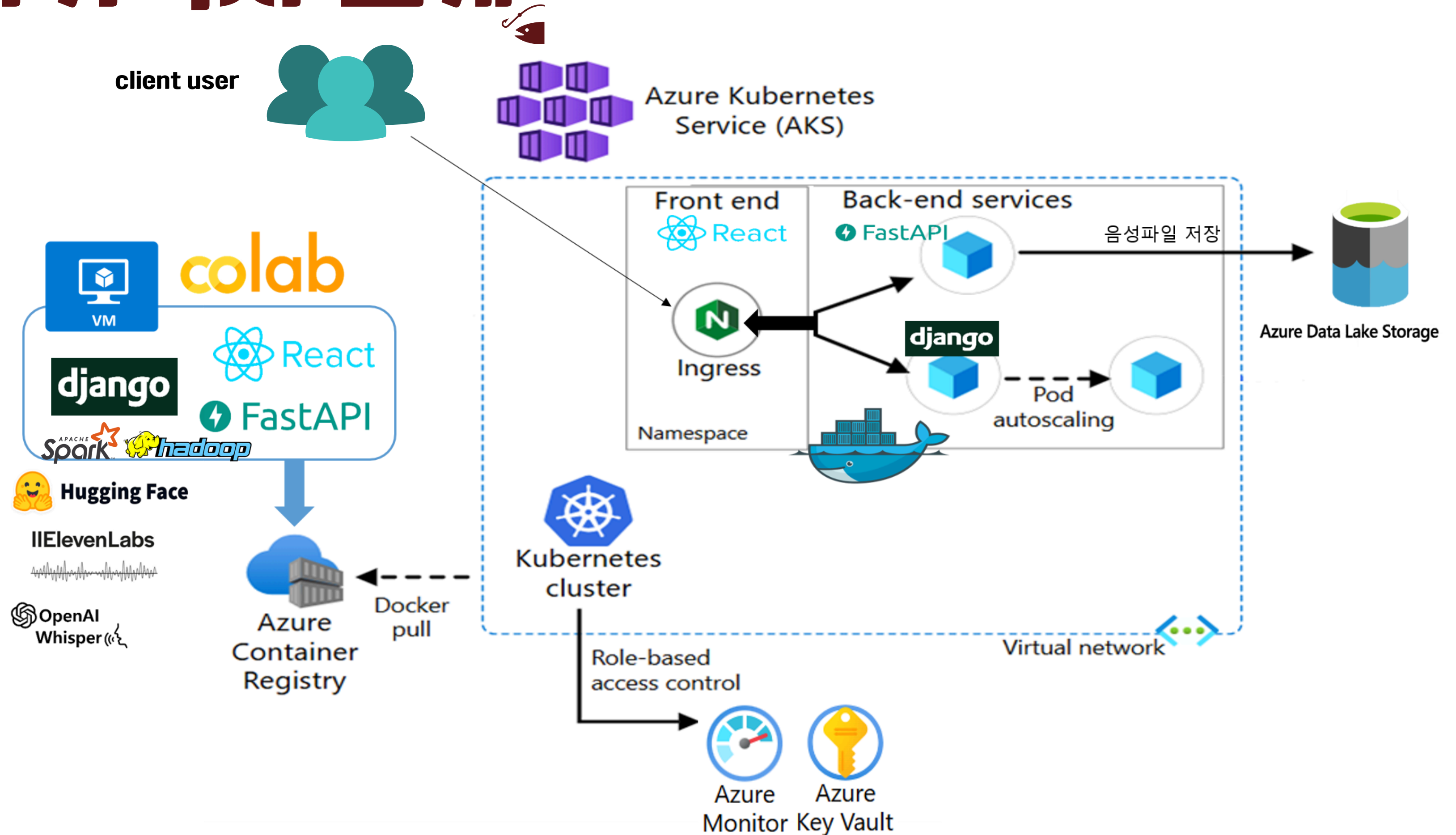


BACK-END

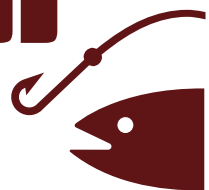
- PYTHON기반의 웹 프레임 워크 사용
 - DJANGO (PYTHON 3.11.11)
 - WEBSOCKET+DAPHNE를 이용한 CHAT BOT (비동기통신) 구현
 - KOLLAMA 모델 연결
 - FASTAPI (PYTHON 3.8.2)
 - 배치처리를 통한 속도 개선
 - AZURE VM을 이용해 DOCKER + KUBERNETES 배포
 - AZURE CLOUD BLOB STORAGE



아키텍처 소개



데이터 소개



KoBERT

보이스피싱 대화 609개와 여행, 음식, 영화, 애완동물 돌보기 등의 일상 대화 609개로 구성된 데이터셋



불용어 처리
토큰화

KoLLaMA

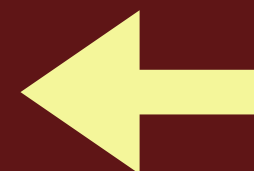
system, assistant, user로 구성된 역할
23개의 보이스피싱 시나리오 데이터셋



epoch와 learning rate를 조절하여
적은 수의 데이터셋 학습

RAG

KBS 사회부가 취재 과정에서 입수한 실제 보이스피싱 시나리오
금융감독원에서 제공하는 보이스피싱 예방요령 자료



롤플레이잉
대응방안 제시



모델링

KoBERT

BertClassifier 구현

- 보이스피싱인지 아닌지 분류하는 이진분류이기 때문에 loss function은 BCEWithLogitsLoss로 설정

KoBERT 모델에 추가학습

- 사전학습된 KoBERT 모델에 정규표현식 텍스트 정제, 불용어 처리, 토큰화 등의 전처리를 한 데이터셋 추가학습

파라미터 선정

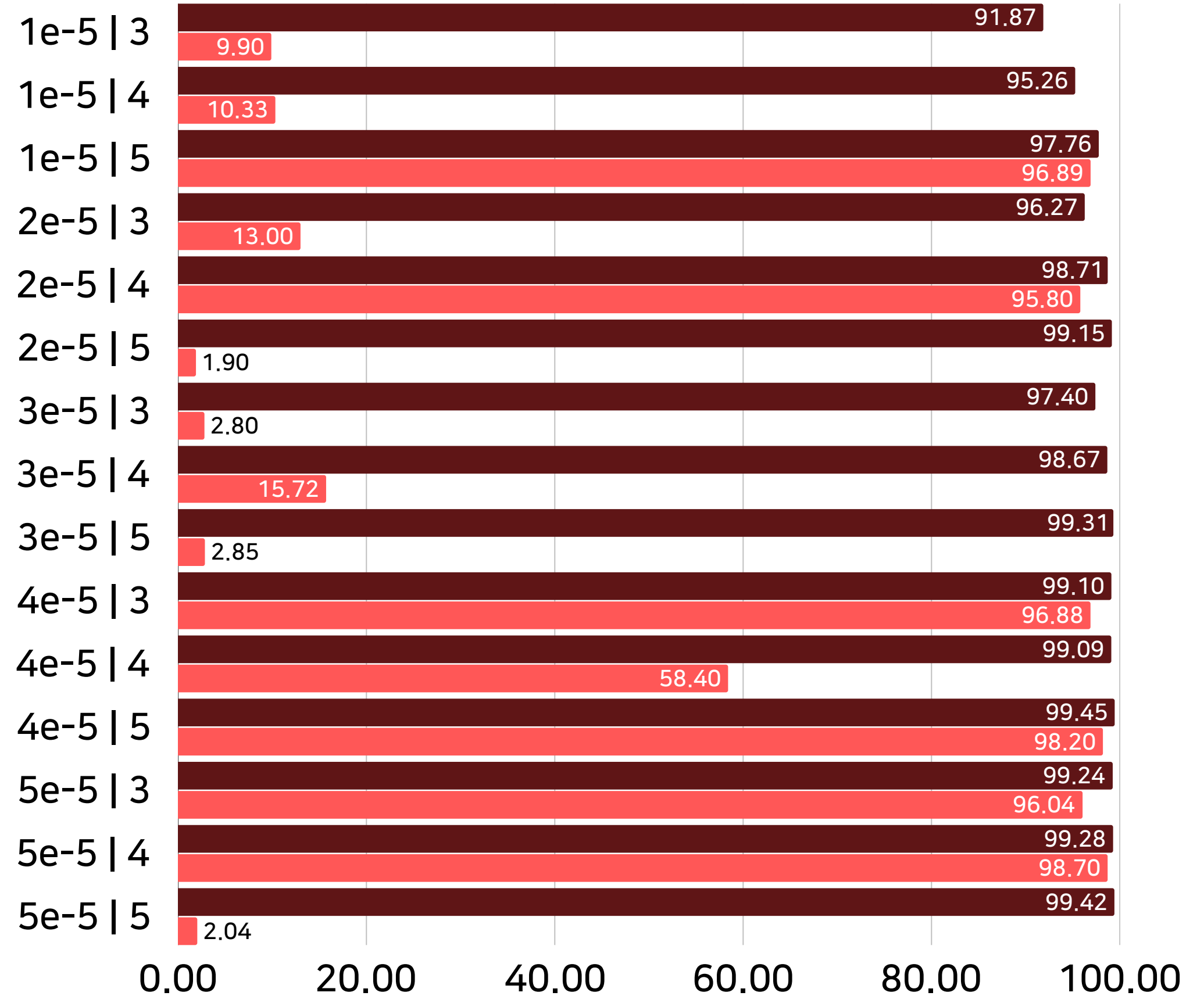
- 학습에 사용된 데이터셋 내의 대화가 아닌 직접 제작한 통화 녹음 파일과 금융감독원의 보이스피싱 녹음 파일을 활용하여 예측값 비교
- learning rate 2e-5, epoch 3 의 파라미터를 최종 선정하였고, max len 64, batch size 16 등의 파라미터는 gpu 메모리에 맞춰 선정

모델 저장

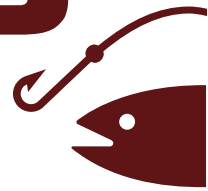
- state dict 형식으로 저장하여 가중치만 KoBERT 모델에 업데이트하는 방식으로 사용

learning rate | epoch

보이스피싱 일상 대화



모델링



KoLLAMA

QLoRA 모델

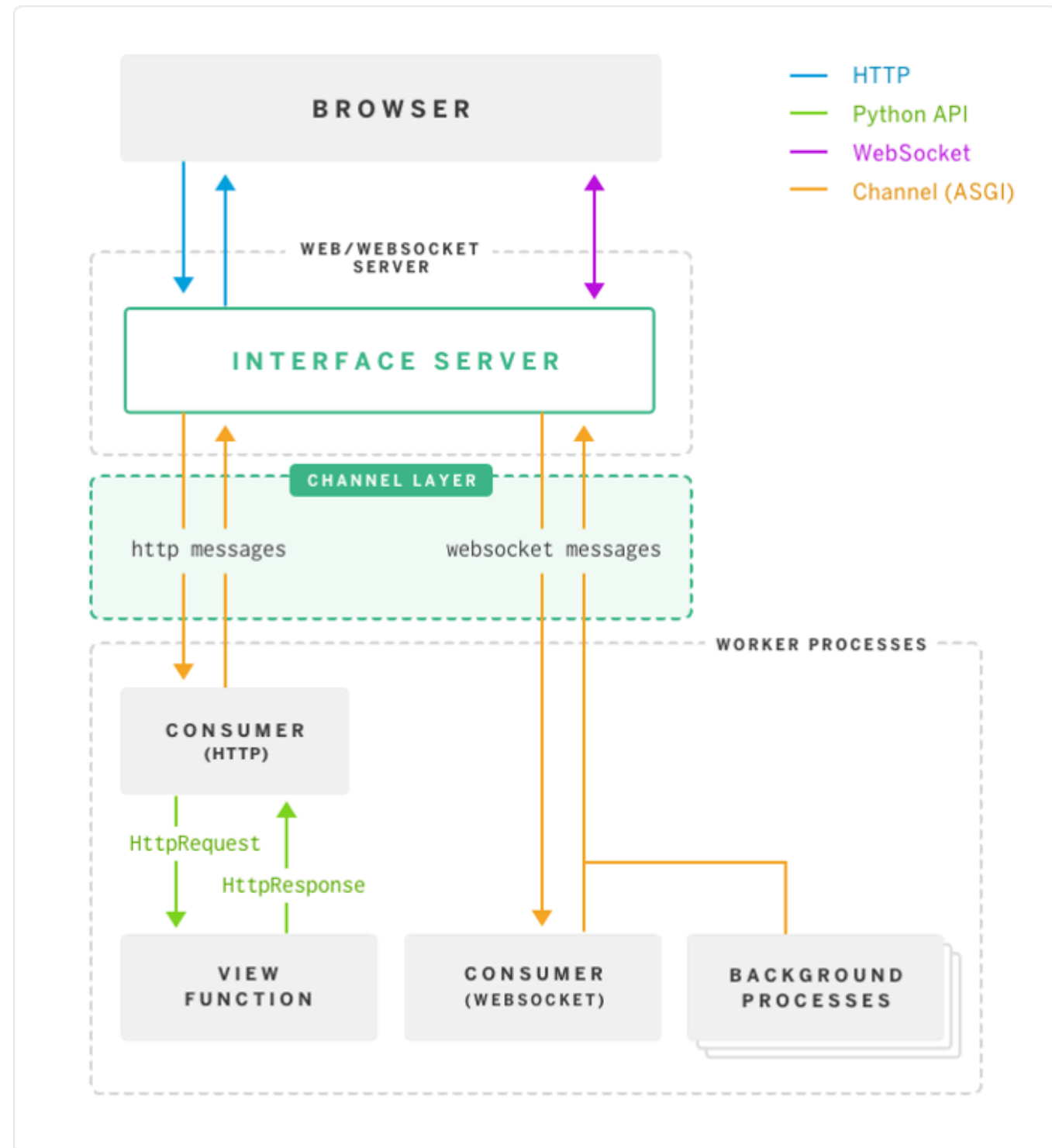
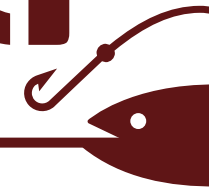
- LLaMA-3-Korean-8B 모델을 4-bit 양자화하여 로드
- 모델 전체 layer의 가중치를 업데이트하는 것이 아니라 LoRA 어댑터 부분의 가중치만 업데이트
- GPU 메모리 사용량을 낮추고 학습 속도를 향상

LangChain과 RAG

- LangChain을 통해 Prompt로 모델에게 보이스피싱 롤플레이팅, 롤플레이팅에 대한 피드백, 보이스피싱 피해 상황에 대한 대응 방법을 제공하도록 설계
- 모델이 응답 제공 시 참고할 수 있도록 RAG를 통해 관련 pdf 문서를 로드하고 쿼리 엔진을 생성하여 response 값 사용
- 학습에 사용되는 데이터셋의 크기가 작기 때문에 Supervised Fine Tuning Trainer(SFTTrainer) 사용
- 작은 데이터셋의 학습을 위해 learning rate 1e-5, epoch 10으로 설정
- LoRA에 대한 파라미터는 rank 4, alpha 16, dropout 0.1로 설정



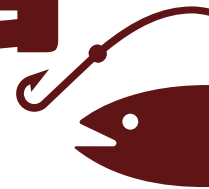
백엔드 채팅 아키텍처



- 기본 DJANGO RUN SERVER는 동기식 처리로 로그인 같은 기능을 처리함
- daphne는 ASGI 웹서버를 통해서 HTTP 요청과 WebSocket 연결을 처리
- WebSocket Consumer에서 웹소켓 연결을 통해 들어오는 메시지를 수신하고, 클라이언트로 모델링의 답변 메시지를 비동기적으로 처리할 수 있도록 함
- 서버가 열리자마자 모델링을 대기시키고 RAG 인덱스를 읽고 초기화 시킨 후 롤플레이팅 화면으로 이동하면 바로 웹소켓 연결하여 모델링의 값을 받아올 수 있도록 클라이언트측에서 기다리는 시간을 최소화시킴
- GPU 메모리 모두 소진시 CPU 동작하도록 설정하고, 만약 모델링 응답이 오래걸려 웹소켓이 타임아웃 되지 않도록 채널에서 타임아웃 설정 및 10초에 한번씩 ping을 보내도록 처리
- 모델링 사전대기 및 코드 모듈화, 배치처리를 통해 기존 모델링 응답 속도 16초 -> 5초로 단축시킴
- 구글 STT와 일래브랩스 TTS를 통해 좀 더 실감나는 롤플레이팅 구현



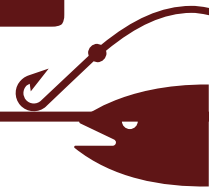
프로젝트 진행 중 발생한 의문점



1. LLM을 추가학습하더라도 추가학습하는 데이터셋의 크기가 충분히 크지 않으면 LLM의 layer 가중치를 크게 업데이트하기 힘들다고 생각한다. 하지만 추가학습 시 보이는 결과의 차이가 꽤 큰데 그 이유는 무엇일까?
2. KoBERT 학습 시 Learning rate 값에 영향을 너무 많이 받는데 그 이유가 무엇일까?
3. KoLLaMA 모델이 추가학습하는 데이터셋에 영향을 덜 받게 하기 위해 Learning rate 값을 많이 줄여보았다. 이런 경우 BASE 모델에 가까운 모델이기 때문에 비정상적인 응답 생성과 같은 문제가 발생하지 않을 것이라고 생각했지만 그렇지 않았다. 그 이유가 무엇일까?



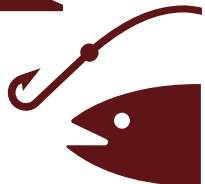
배포 웹사이트 시연



배포 웹사이트 시연 영상





Q & A 



LLM을 활용한

THANK YOU

보이스피싱 예방
웹 어플리케이션
프로젝트

