보안 매뉴얼 파인튜닝 모델 개발 보고서

책임 작성자: 김경수

과정 1) 보안 매뉴얼 -> 허깅페이스 데이터 셋으로 변환

PDF 파일인 보안 매뉴얼을 데이터 셋 형식으로 바꾸는 과정이 필요하다.

우선 복잡한 구조로 되어있는 매뉴얼 파일을 랭체인과 ChatGPT API를 통해 input과 output으로 구성된 JSON 파일로 변환한다.

이때, 프롬프트 엔지니어링을 통해 데이터 증강 기법을 적용하여 14개의 상세 매뉴얼에 대해 200가지 질문-응답 구조로 구성된 JSON 데이터로 변환한다.

과정 2) 허깅페이스 데이터 셋 배포

JSON으로 변환한 데이터를 허깅페이스 데이터 셋 양식에 맞게 변환한다.

허깅페이스에 데이터 셋 레포지터리(kingkim/DS_Building_SecurityManual_V5)를 생성하고 모델을 배포한다. (https://huggingface.co/datasets/kingkim/DS_Building_SecurityManual_V5)

과정 3) 파인튜닝에 적합한 모델 찾기

매뉴얼 챗봇 구현을 위해 파인튜닝을 하는 것이기에, 반드시 로컬의 CPU 환경에서도 구동 가능한 모델을 찾아야 했다.

다양한 언어 모델(GPT, BERT)들의 구동 조건 및 환경을 조사해본 결과, GPT-2 기반의 모델이 기본적으로 CPU 환경에서도 잘 구동된다는 사실을 알았다.

처음에는 한국어가 잘 학습된 SKT의 KoGPT 모델을 선택하려 했으나, 미세 조정 하려는 데이터 양에 비해 모델의 규모가 너무 컸다. 즉, 과적합(overfitting) 문제로 인해 보다 가벼운 규모의 모델을 찾아야 했다.

우리 프로젝트에 조건에 부합하는 GPT-2 모델에는 DialoGPT와 DistilloGPT가 있었다.

이 중에서 한국어가 잘 학습되어 있는 모델은 'heegyu/kodialogpt-v1'였다. 마이크로소프트에서 공개한 GPT-2 모델인 DialoGPT에 SKT에서 공개한 한국어 대화 데이터 셋을 학습시킨 모델이었다.

과정 4) 파인 튜닝

아래의 과정을 통해 구글 코랩 환경에서 무료 GPU(T4) 사용해서 선택한 모델을 미세조정 한다.

1단계) 튜닝 할 모델, 토크나이저, 데이터 셋 불러오기

2단계) 프롬프트 생성 함수와 데이터 셋 전처리

3단계) 학습 설정

4단계) Trainer 설정

5단계) 모델 파인튜닝

과정 5) 파인튜닝 모델 허깅페이스 배포

구글 코랩의 무료 GPU(T4) 환경에서 파인튜닝한 모델을 다음과 같은 허깅페이스의 Model 레포지터리에 업로드 한다. (kingkim/kodialogpt_v3.0_SecurityManual)

(https://huggingface.co/kingkim/kodialogpt_v3.0_SecurityManual)

과정 6) 직접 파인튜닝한 모델을 바탕으로 로컬의 CPU 환경에서 사용 가능한 챗봇 만들기

파인튜닝 모델: kingkim/kodialogpt_v3.0_SecurityManual (Base model: heegyu/kodialogpt-v1)

1단계) 데이터 전처리: PDF 보안 매뉴얼을 텍스트로 변환한 후, 문장 단위로 분리합니다.

2단계) 문장 인코딩: SentenceTransformer를 사용해 각 문장을 벡터로 변환합니다. 이 벡터는 ChromaDB에 저장될 거예요.

3단계) 데이터 저장: 변환된 벡터를 ChromaDB에 저장합니다. 이때, 원본 문장과 해당 벡터를 함께 저장해 나중에 참조할 수 있도록 합니다.

4단계) 질문 처리: 사용자가 질문을 입력하면, 이 질문을 SentenceTransformer로 인코딩합니다.

5단계) 유사도 검색: 질문 벡터와 ChromaDB에 저장된 문장 벡터들 간의 유사도를 계산합니다. 가장 유사한 문장을 추출하죠.

6단계) GPT-2로 답변 생성: 가장 유사한 문장을 Hugging Face의 GPT-2 모델에 입력해 자연스럽게 답변을 생성합니다.

과정 7) 배운 점 및 발전 방향

그동안 Gemini, Llama, OpenAI 등을 활용해 잘 완성된 모델을 API를 호출해서 애플리케이션을 만들어왔다. 가장 빠르고 효율적이며 높은 성능을 구현할 수 있는 방법이라 생각했다.

하지만, 얼마 전 문득 다음과 같은 생각을 하게 되었다.

모든 회사가 최고의 성능이 구현된 지능형 애플리케이션이 필요할까? 성능은 조금 떨어지더라도 안정성이 높은 AI 모델이 필요한 회사, AI 모델이 회사의 기밀을 담아야 하기에 API 호출이 불가능한 상황인 회사 등 다양한 이유로 자체적인 AI 모델이 필요한 회사도 반드시 생겨날 것이라는 생각을 했다.

결국 공개된 AI 모델과 API를 호출해서 잘 활용하는 역량도 중요하지만, 기존의 공개 모델을 미세조정을 통해 최적화된 모델을 만들 수 있는 역량도 필요하다는 생각이 들었다.

이러한 이유로 이번 프로젝트를 통해 데이터 셋부터 시작해 독자적인 모델까지 직접 구축해보자는 생각을 하게 되었다.

무엇보다 허깅페이스 환경을 적극적으로 활용해보았고 레포지터리를 만들어 데이터 셋과 파인튜닝한 모델을 직접 배포까지 해 보는 과정에서 많은 것을 배우고 시야를 넓힐 수 있었다.

그 과정에서 다음 성장을 위해 반드시 해야 할 미션도 찾았다.

모델을 학습시키는 것보다 가장 힘들었던 것은 모델에 적합한 데이터를 생성하는 것이었다. 데이터를 여러 기법을 활용하고 증강하는 것, 그리고 정규화하는 것이 정말 중요하다는 것을 알았다.

이 과정을 보다 원활하게 진행할 수 있는 알고리즘은 어떤 것이 있을지 보다 많은 고민이 필요하다. 특히 프로젝트에선 데이터 증강을 랭체인과 OpenAI 호출을 활용했지만 더 좋은 방법을 고민해볼 필요가 있다.

마지막으로 파인튜닝 과정에서 이번 프로젝트에서 사용한 방법보다 보다 효율적이고 좋은 성능을 가질 수 있게 하는 방법이 있다는 것을 알았다. 바로 QLoRA(Quantized Low-Rank Adaptation)를 통해 GPU 사용량을 감소시키는 것과 동시에 모델 크기를 줄이면서도 높은 성능을 유지할 수 있도록 하는 것이다.