[7팀 모듈프로젝트 발표 대본]

[1 pages | 팀 프로젝트 소개]

안녕하십니까, 피싱 가드 챗봇 개발 프로젝트를 진행한 7팀 BlackList입니다.

[2 pages | 목차 설명]

발표 구성입니다. 프로젝트 개요 -> 프로젝트 팀 구성 및 역할 -> 수행 절차 및 방법과 수행 경과와 자체 평가 의견 순서로 발표하겠습니다.

[3 pages | 프로젝트 개요/배경]

최근 피싱 사이트의 수법이 날로 교묘해지고 있습니다. 특히 URL을 기반으로 한

피싱 공격은, 사용자가 짧은 순간에 잘못된 사이트에 접속하게 만드는 만큼 보안상 큰 위협으로 꼽히고 있습니다. 따라서 저희는 이를 예방하기 위해 AI 기반 악성 URL 탐지 모델을 직접 학습시켜 대안 웹 사이트 제공해주는 챗봇 시스템 주제를 계획하게 되었습니다. 사용자가 입력한 URL의 피싱 여부를 AI 어시스턴트가 분석을 합니다. 그 후 블랙리스트에 URL이 존재하면 바로 챗봇이 위험한 사이트라고 알려줍니다. URL이 존재하지 않으면 학습한 머신러닝 모델로 예측하여 악성 사이트일 경우 블랙리스트에 추가 후 AI 보안 어시스턴트가 대안 사이트 제공하는 답변을 생성해줍니다. 이를 통해 피싱 사고 예방과 보안 관련 지식이 없어도 악성 사이트를 쉽게 판단할 수 있습니다. 그리고 관련 상담이나 지식을 습득하는 것도 가능합니다.

[pages 4-5 | 팀 구성 및 역할, 수행 절차 및 방법]

ppt 흐름 읽는 방식, 예) 기획서 작성, 팀 역할 분담, 일정 수립 등 사전 기획 데이터 수집을 화요일에

[ | 프로젝트 팀 구성 및 역할]

구현 팀 -> 기능 구현 내용

[ | 프로젝트 수행 및 방법]

수행하였습니다.

이 2개는 ppt보면서 읽기!

[**6 pages** | 프로젝트 수행 경과]

첫번째 Dataset 구성, 모델 선정 데이터 전처리 및 머신러닝을 통한 데이터 기반 모델 학습을 했습니다. 저희는 XGBoost 모델을 선정하여 모델의 정확도를 77%까지 올렸습니다.

두번째 response api 기반 통합 구조를 수행하였습니다. 구체적으로는 블랙리스트, function\_call, file\_search등을 통해 AI 어시스턴트를 구현하였습니다.

다음으로 UI 구성을 수행하였습니다. Streamlit을 활용하여 사용자 친화적인 대화창을 제공해주어 사용자 편의성을 갖추었습니다.

마지막으로 디버깅 및 리팩토링을 진행하면서 발표준비를 하였습니다.

[논문 참고하여 트리 구조 모델 중 Colab Pro에서 사용할 수 있는 GPU와 가장 합이 좋은 XGBoost 모델]

[**7 pages** | 피싱 URL 탐지 모델 부분]

(모델 선정 과정) 기존에 Random Forest 모델을 사용하였으나 GPU 지원 기능이 없어 학습시간이 오래걸리는 문제가 생겼습니다. 그래서 한국정보통신학회에 있는 악성 URL 머신러닝 탐지모델 개발 논문을 참고하여 같은 트리 구조 모델중 GPU 사용을 지원하여 Colab Pro와 가장 합이 좋은 XGboost로 변경하게 되었습니다

XGBoost는 경사하강법을 활용하는 지도 학습 부스팅 알고리즘인 그레디언트 부스트 결정 트리를 사용하는 앙상블 머신 러닝 모델입니다.

그레디언트 부스팅이란 여러개의 얕은 트리를 두어 순차적으로 강화시켜 이전 트리의 실수를 다음 트리가 보완해주는 방식입니다.

즉, 첫번째 트리의 오차를 두번째 트리가 보완하고 두번째 트리의 오차를 세번째 트리가 보완해주는 방식으로, 계속 반복해서 그레디언트를 줄이는 방향으로 학습이 이루어집니다.

XGboost는 과적합 방지, 정확도, 속도를 개선할 수 있고 회귀 및 분류 문제에서 모두 사용이 가능합니다. 대규모 Dataset에서도 뛰어난 성능을 발휘합니다.

또한 병렬 처리가 가능하여 빠른 학습이 가능합니다.

[**8-9 page** | 데이터 정규화, 불균형 문제 해결, 해결 결과]

새로 준비한 데이터셋에 정상 URL은 대부분 http나 https 프로토콜이 없이 도메인만 있고

악성 URL은 대부분 https가 붙은 상태로 존재하여 모델이 정확한 탐지에 어려움이 있었습니다. 그래서 저희는 데이터 정규화를 통해 관련 프로토콜을 feature에서 제거하여 모델이 URL의 실제 구조적 특성에 집중할 수 있도록 했습니다.

또한 새로 준비한 약 65만개 데이터 셋에 악성 정상 데이터 비율이 2대 1로 불균형하게 존재하여 데이터 전처리를 통해 데이터 개수 비율을 정상과 악성 1대1로 맞추어 약 40만개 데이터를 새로운 CSV 파일로 생성했습니다.

결과는 이렇게 나옵니다~~

[**10 page** | Feature 선정, 근거]

다음은 Feature 선정 및 근거입니다. 악성 URL은 도메인을 감추기 위해 숫자를 섞거나 경로를 불필요하게 늘려 전체 문자열 길이를 길게 만드는 경우가 많습니다. 또한, 공식 사이트들은 IP 주소 대신 도메인 이름을 사용하기 때문에, IP 주소가 포함되어 있는지도 중요한 판단 기준이 됩니다.

피싱 공격자는 리디렉션이나 쿼리 문자열을 통해 사용자를 속이는 경우가 많아, 특수문자의 사용 빈도가 높으며, @기호를 이용해 URL 앞부분을 무시하게 만드는 수법도 자주 사용됩니다.

이러한 특징들을 바탕으로 모델에서 사용될 feature를 URL의 전체 문자열의 길이, URL의 경로 길이, IP주소 포함 여부, URL 내 점(.)의 개수, 특수문자 수, @ 기호 포함 여부, 숫자개수 로 총 7개의 Feature를 선정하였습니다.

위 Feature 선정은 캘리포니아 어바인대학에서 운영하는 머신러닝 사이트 자료를 바탕으로 선정했습니다.

[**11 page** | 학습 데이터 및 테스트 데이터 분할, XGBoost 모델 생성 및 학습, 성능 평가]

다음 보시는 코드는 학습 데이터 및 테스트 데이터를 분할하고 XGBoost 모델 생성 및 학습하여 성능 평가 후 모델을 pickle 파일로 저장하는 코드입니다. 앞에서 설명드린 Feature로 추출한 데이터를 8 : 2 비율로 학습 및 테스트 데이터로 나누어 XGB 모델에 학습시켰습니다. 모델 생성 초기에 데이터 과적합이 발생하여 max\_depth와 learning\_rate의 하이퍼 파라미터 값을 4와 0.5로 조정하였더니 과적합 문제가 해결되었습니다.

[**12 page** | xgb 모델 성능 평가]

모델 전체 정확도는 77%로 학습시킨 데이터에서는 높은 평가를 보입니다.

[**13 page** | 전체 시스템 작동 로직 흐름도]

사용자가 입력한 URL의 피싱 여부를 AI 어시스턴트가 분석을 합니다. 입력이 URL과 관련이 없으면 OpenAI가 답변을 하지만 관련이 있고 피싱 URL 경우 Function call 기능이 동작해 블랙리스트를 확인합니다. 블랙리스트에 존재하는 URL이면 모델 예측 없이 바로 위험한 사이트 대화 내용과 파일 서치 분석 내용을 추가로 제공하고 블랙리스트에 저장됩니다. 하지만 URL이 존재하지 않으면 학습한 머신러닝 모델로 예측하여 악성 사이트일 경우 블랙리스트에 추가 후 AI 보안 어시스턴트가 대안 사이트 제공하는 답변을 생성해줍니다.

[page 14, 메인 에이전트가 호출하는 함수에 대한 내용]

보시는 코드는 메인 에이전트가 호출하는 함수입니다. 여기서 호출되는 새로운

Response 객체는 악성 url과 악성 url 탐지에 대한 전문적인 내용이 담겨져있는 pdf를 vector\_store(악성 url에 대한 감지하는 정보)로 받게 되어 보다 전문적인 대답을 메인 에이전트에게 전달하게 됩니다.

실행 로직은 다음과 같습니다. Check\_black\_list 함수를 통해 블랙리스트 확인 및 XGBoost 모델 결과를 반환받게 되며 해당 결과가 response 객체에 전달되어 보다 풍부한 내용을 메인 에이전트에게 전달하게 됩니다.

[page 15, 메인 에이전트가 사용하는 tool에대한 설명]

다음은 메인 에이전트 클래스의 tool 부분입니다. 메인 에이전트는 이전 페이지에서 보신 agent\_call 함수를 필요에따라 호출하게 됩니다. 해당 함수는 메인 에이전트가 사용자와 대화하다가 악성 url관련 질문을 받으면 함수를 호출하게 description을 적었습니다.

또한 사용자와의 대화를 담당하는 메인이기 때문에 websearch를 툴로 사용하며 이는 일반적인 사용자와의 대화에도 사용될 수 있고, 악성 url로인한 대안사이트를 찾는데도 사용될 것입니다.

[page 16, 블랙리스트 로직 및 모델 결과 반환 함수에 대한 설명]

다음 보시는 코드는 블랙리스트 확인과 XGBoost 모델을 호출하는 함수입니다.

먼저 url을 받게되면 블랙리스트에 있는지 확인하고 블랙리스트에 존재한다면 바로 return을 호출하여 반환합니다. 만약 블랙리스트에 없다면 url을 각 feature로 나누어 전처리를 하고 모델에 넣어 값을 반환받습니다. 이 과정은 model\_call 함수 내부에서 전부 실행됩니다. 모델 결과 정상 url로 판단되면 return을 호출하여 결과를 반환하고, 악성 url로 판단되면 블랙리스트에 새로 넣고 csv로 저장하여 갱신하며 마찬가지로 return을 호출하여 결과를 반환합니다.

[page 17, 예시 결과 화면에 대한 설명]

다음은 예시 결과화면입니다. Streamlit으로 UI를 구성하였으며 다음과 같이 대화가 진행됩니다. 더 자세한 내용은 이후 시연 영상에서 보여드리겠습니다.

[page 18, 블랙리스트 UI 화면에 대한 설명]

다음으로 local 상의 blacklist.csv를 불러 다음과 같이 streamlit 화면으로 보이게 구성한 화면입니다. 사용자는 해당 탭을 통해 blacklist를 직접 볼 수 있습니다.

[page 19, 프로젝트 요구사항 충족에 관한 내용]

다음으로 프로젝트 요구사항 충족에 관한 내용입니다.

저희는 OpenAI API에서 제공하는 Responses API를 기반으로, websearch 및 file search를 사용하였습니다. 그리고 커스텀 함수를 구현하여 직접 학습시킨 모델을 .pkl로 저장하여 AI agent가 사용할 수 있게 구현하였고, 적절한 시스템 프롬프트를 구현하여 대안 사이트등 에이전트가 적절하게 답을 할 수 있도록 구조를 명확화하였습니다. 또한 사용자와의 메인 대화를 당담하는 상위 에이전트, 블랙리스트 검사, 악성코드 분석 및 리포팅을 전달하는 하위 에이전트로 구성하여 멀티 에이전트 구조를 구현해보았습니다. 마지막으로 Streamlit을 이용해 간단하게 사용자 친화 UI를 제공하여 사용자가 보다 쉽게 사용할 수 있도록 구성하였습니다.

[page 20 | 심화 요구사항 충족]

다음은 심화 요구사항 충족입니다.

저희는 악성 url 탐지 관련 보고서를 통한 파일 분석 반영 결과를 통해 자동 생성 요구사항을 만족하였고

Open API의 session state 기능을 통해 이전 대화를 저장하여 사용자별 맞춤형 에이전트를 구현하였습니다. 그리고 블랙리스트 csv 파일에 악성 url을 저장하여 실시간으로 로깅에 대응하였습니다. 사용자에게 전달받은 입력에 URL이 존재하였을때 그 URL의 정보나 악성 URL일 경우 대체 사이트를 추천하는 시나리오를 추가하였습니다.

[page 21 | 시연 영상]

다음은 시연 영상입니다. // **시연 영상하면서 가볍게 코멘트 던져주기**

[page 22 | 자체 평가 의견]

마지막으로 자체 평가 의견입니다. 첫번째, 프로젝트 결과물 완성도 평가 별점 5점입니다.

계획안 작성 초기에 악성 URL을 사용자에게 받으면 대안 사이트를 추천하는 기능과 블랙리스트에 악성 URL 추가하는 기능을 모두 구현하여 5점을 주었습니다.

두번째, 팀의 칭찬할 점과 아쉬운 점입니다. 저희 팀은 프로젝트 진행하면서 단 한명의 무임승차도 없고 각자 맡은 역할을 성실히 수행해주셔서 최상의 팀워크라고 단언컨데 자랑 할 수 있습니다. 평가 의견 적을 때 아쉬운 점을 정하기가 정말 어려웠는데, 그래도 하나를 말하자면 맨 처음 주제 선정시 낯가림을 선정할 수 있겠습니다.

세번째, 프로젝트의 시간을

[Q&A]

1. 왜 블랙리스트에 먼저 확인하나요?

(답변) 속도 향상을 위해~~~

피싱 사이트인 경우 클릭 한 번으로 사고가 일어나기 때문에 즉각적인 대응과 솔루션이 중요하다고 생각합니다. 그래서 모델에 입력받아 예측받는 결과를 받는 시간보다 블랙리스트에 있는 경우 바로

1. 시연 영상에서 가져오신 URL은 진짜 악성 URL인가요? 어디서 가져오셨나요?

(답변) page 23 참고자료

영상에서 본 URL은 kaggle 악성 URL 데이터에서 가져온 악성 URL데이터 입니다.

1. Feature 개수를 왜 더 늘리진 않아보셨나요?

모델 개발 첫날에 Whois 라이브러리를 DNS를 사용하여

Vector store 부분 하위 에이전트에

피드백:

1. 자체평가 의견 더 자신있게
2. 9페이지 모델 평가표

조금 더 자신감 있게 장점

시연영상 -> 간단한 설명