404 Not Found

매크로 탐지 확장 프로그램

김송혜, 김진섭, 송일환, 최서연

목 차

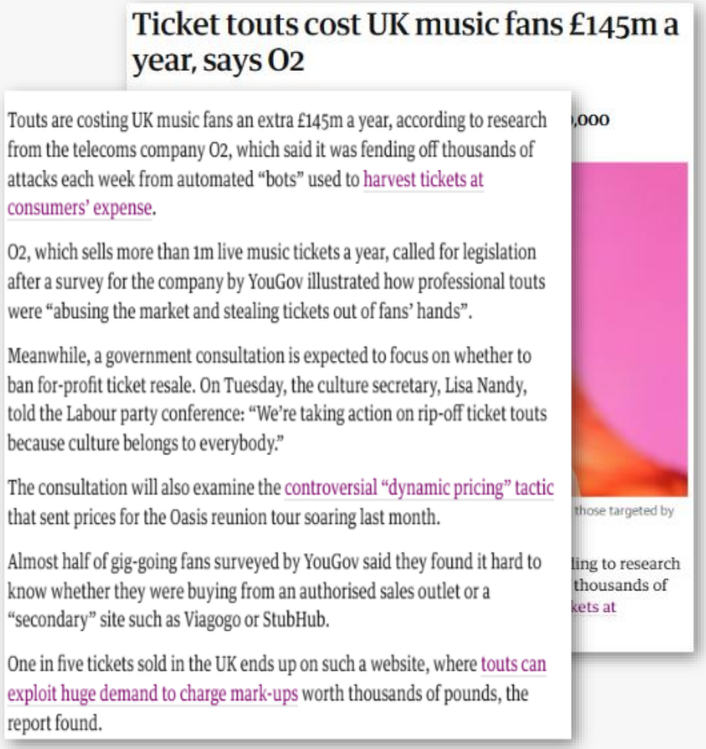
01

배경

매크로(Macro)란?

매크로는 사람이 직접 수행해야 하는
클릭, 입력, 이동 등의 반복 작업을 **자동**으로 실행하도록
만든 프로그램 또는 스크립트이다.
키보드나 마우스의 동작속도 등을 조절할 수 있다.

매크로로 인한 피해 사례



Ticket touts cost UK music fans £145m a year, says O2

Touts are costing UK music fans an extra £145m a year, according to research from the telecoms company O2, which said it was fending off thousands of attacks each week from automated "bots" used to harvest tickets at consumers' expense.

O2, which sells more than 1m live music tickets a year, called for legislation after a survey for the company by YouGov illustrated how professional touts were "abusing the market and stealing tickets out of fans' hands".

Meanwhile, a government consultation is expected to focus on whether to ban for-profit ticket resale. On Tuesday, the culture secretary, Lisa Nandy, told the Labour party conference: "We're taking action on rip-off ticket touts because culture belongs to everybody."

The consultation will also examine the controversial "dynamic pricing" tactic that sent prices for the Oasis reunion tour soaring last month.

Almost half of gig-going fans surveyed by YouGov said they found it hard to know whether they were buying from an authorised sales outlet or a "secondary" site such as Viagogo or StubHub.

One in five tickets sold in the UK ends up on such a website, where touts can exploit huge demand to charge mark-ups worth thousands of pounds, the report found.

Ticketmaster가 테일러 스위프트 콘서트 예매 중
'매크로 봇 공격'으로 서버가 마비되고,
수억 건의 자동 요청으로 취소되는 사태가 발생

https://www.cbsnews.com/news/taylor-swift-fans-battle-ticket-bots-and-ticketmaster

사회 > 전국

매크로 이용 온라인 암표판매 피의자 대구

【파이낸셜뉴스 대구=김장욱 기자】 매크로를 이용해 온라인 암표를 판매한 피의자가 경찰에 올해 처음으로 붙잡혔다.

대구경찰청 사이버범죄수사대는 매크로 프로그램을 이용해 야구경기 입장권을 예매 후 부정판매한 혐의(국민체육진흥법)로 A씨(40대)를 검거해 수사 중이다.

경찰에 따르면 A씨는 티켓 예매 사이트에 접속, '지정된 명령을 자동으로 반복 입력하는 매크로프로그램'을 이용해 삼성라이온즈 홈경기 티켓 1매를 9000원에 예매하고, 이를 티켓 판매 사이트에서 1만5000원에 부정판매한 혐의다.

또 같은 방법으로 삼성라이온즈 홈경기 티켓 총 133매를 예매해 120회에 걸쳐 241만원 상당을 부정판매한 것으로 확인됐다.

A씨 외 매크로 프로그램을 이용한 다른 의심거래 건에 대해서도 단서를 확보하고 수사를 확대할 계획이다.

매크로 프로그램으로 야구 경기 티켓을 대량 구매 후
120회 이상 재판매한 피의자가 형사처벌된 사례

https://www.fnnews.com/news/202507310824490070

매크로로 인한 피해 사례

수십 년이 지난 지금도 별반 다르지 않다. 지난달 출시된 넷마블의 한 게임에서는 불법 프로그램 사용으로 차단되는 계정만 하루에 수만 개에 달한다. 한 블록체인 게임은 매크로로 재화를 벌고 빠르게 가상화폐로 환전한 뒤 자취를 감추는 이용자들로 인해 골머리를 앓고 있다.

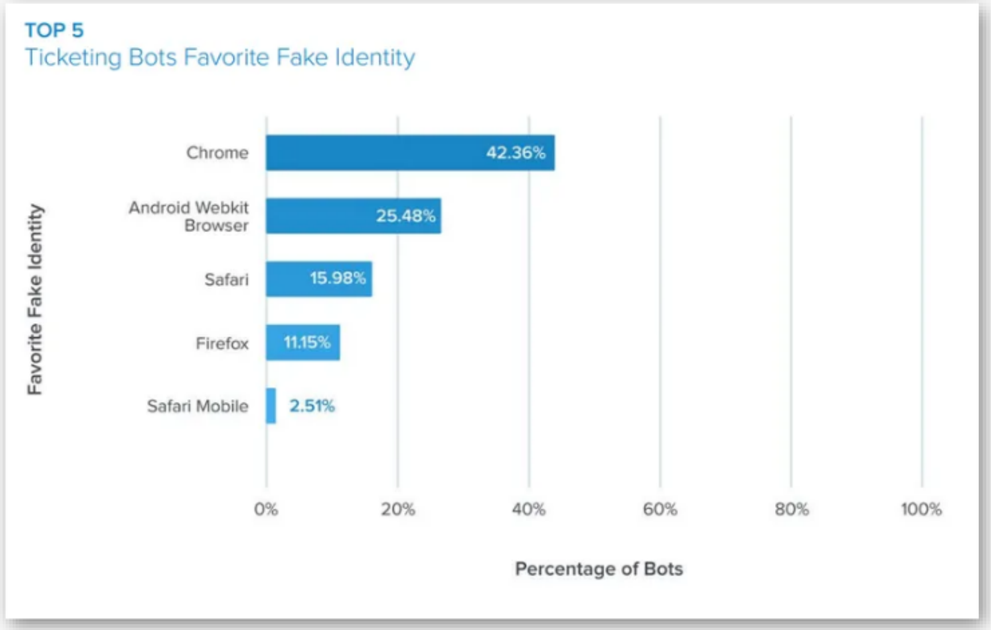
매크로를 악용해 발생하는 문제는 게임에만 국한되지 않는다. 인기 가수의 콘서트나 스포츠 경기를 예매할 때는 매크로를 활용해 먼저 티켓을 차지하고 웃돈을 얹어 판매하는 일이 비일비재하다.

다만 게임과 다른 점이 있다면 매크로를 써서 티켓을 구입해 부정 판매하는 행위는 적발 시 처벌 대상이 된다는 점이다.

반대로 게임 내에서 매크로 프로그램을 사용하는 것은 처벌할 근거가 마땅치 않다. 현행법에서는 게임 내 불법 프로그램을 제작·유포하는 행위는 처벌하지만, 이용하는 행위에 대해서는 별도의 금지 규정을 두지 않고 있기 때문이다.

게임사의 대응이 계정 정지 수준에 그칠 수밖에 없는 이유다. 이용자에 대해서도 제재를 가하는 법안이 지난해 발의됐지만 1년이 지난 지금도 감감무소식이다.

게임은 K콘텐츠 열풍을 이끄는 숨은 효자다. 올해 상반기 게임산업은 3조원이 넘는 흑자를 거뒀다. 음악산업의 거의 4배다.



매크로 문제는 ‘개별 이용자 부정행위’에 그치지 않고,
게임 내 경제 붕괴, 티켓 예매 시장 왜곡, 가상자산 환전 악용 등
다방면에서 피해 발생

티켓 예매 봇들이 가장 많이 위장하는 브라우저 유형

https://v.daum.net/v/20250925173003602

How Bots Affect Ticketing (Imperva 기반 보고서)

기존 웹서비스 매크로 차단 시도

- 계정·수량 제한(1인당 예매수 제한) 및 거래 제재
- CAPTCHA / 자동문자(자동문자열) 차단
- IP/접속 패턴 모니터링·차단, 쓰로틀링
- 행동 패턴 분석(비정상적 클릭/타이밍 탐지)
- 서버·API 레벨 보호 (비정상 요청 검증 등)

…같은 '암표'…인터파크티켓이 부정예매 막는 …

| 임경영 CTO "얼굴인식 기술·취소표 대기시스템 도입 예정"

인터넷 | 입력 :2024/09/01 12:18 수정: 2024/09/01 20:34

안희정 기자 | 기자 페이지 구독 기자의 다른기사 보기

최근 들어 콘서트·뮤지컬 등 공연 티켓을 예매하는 게 더 어려워진 느낌이다. K팝 인기가 날이 갈수록 커지며 티켓팅 경쟁이 국내에 한정되지 않고 글로벌하게 더 치열해졌기 때문일까. 예매 전부터 머릿속으로 시뮬레이션도 돌려보고 티켓 쟁탈전에 임하지만 만만치 않다. 취소표를 구하는 취켓팅도 힘들어졌다.

그럼에도 불구하고 누군가는 티켓 구매에 성공한다. 빠르게 움직인 결과이기도 하지만, 간혹 불법적인 기술(?)이 들어가기도 한다. 이렇게 구매한 티켓을 웃돈을 받고 다시 판매하는 업자들도 있다. 암표상들인 것이다.

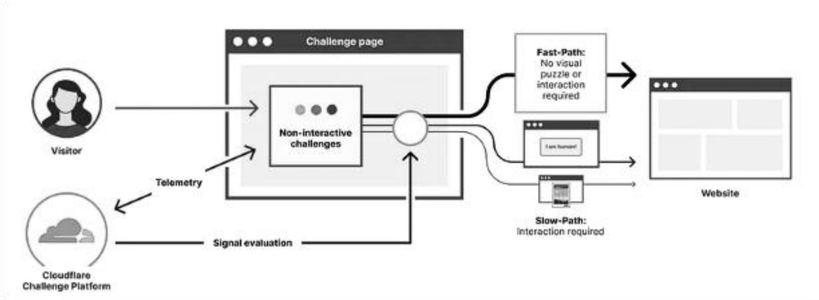
정부도 심각성을 깨닫고 공연과 스포츠 분야 암표 근절 정책을 강화하고 나섰다. 올해 3월부터 매크로 프로그램으로 공연 입장권을 구입해 부정 판매했다가 걸리면 1년 이하의 징역이나 1천만원 이하의 벌금을 내야 한다.

다만 매크로 프로그램을 티켓 판매 플랫폼들이 알아차리기는 쉽지 않다. 프로그램을 써서 구매한 건지 아

기존 서비스들

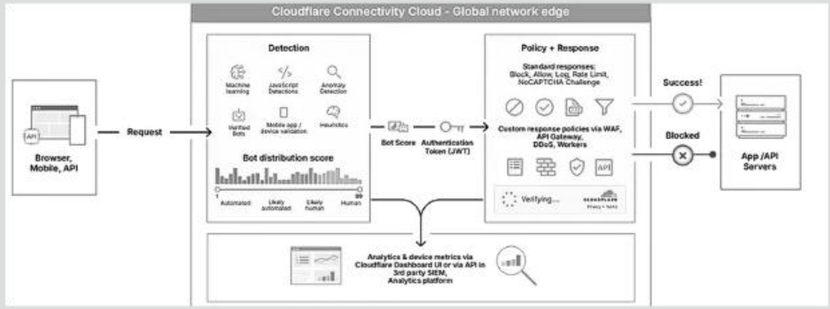
CAPTCHA

- 사용자가 "나는 로봇이 아닙니다" 버튼을 클릭
- 이미지 퍼즐을 푸는 인증 방식

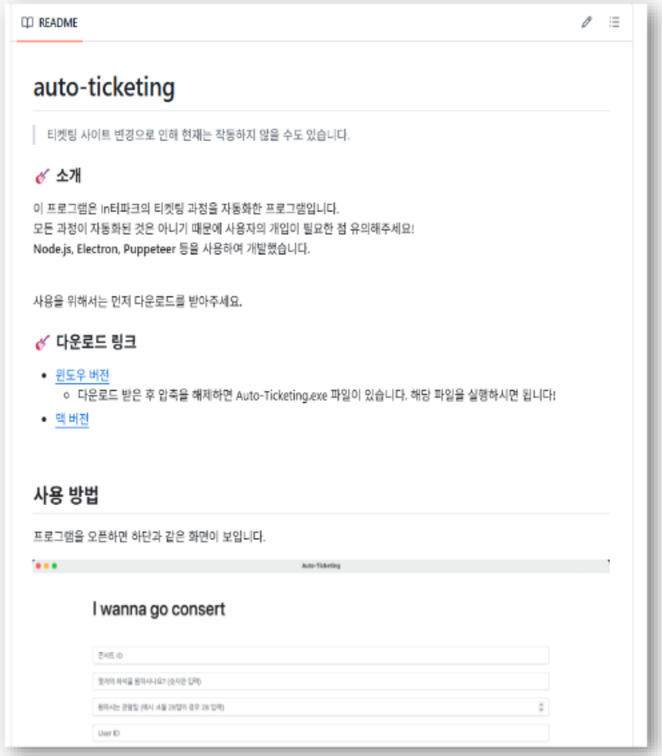


Bot Management

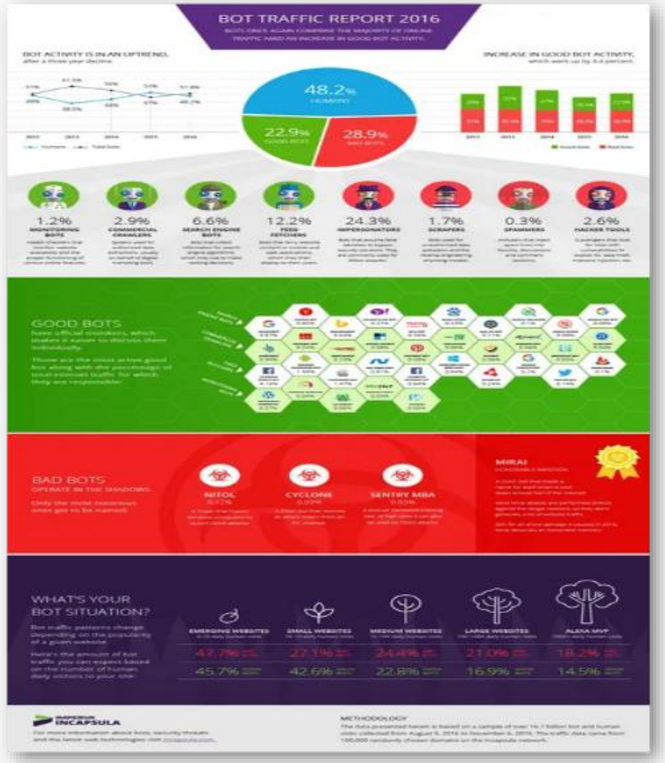
- 요청의 헤더, 쿠키, 행동 패턴등을 분석
- 머신러닝 기반 Bot Score(위험 점수) 로 자동 판단



그럼에도 매크로 막는 것은 어렵다



깃허브 및 여러 SNS에서도
쉽게 매크로를 구할 수 있다



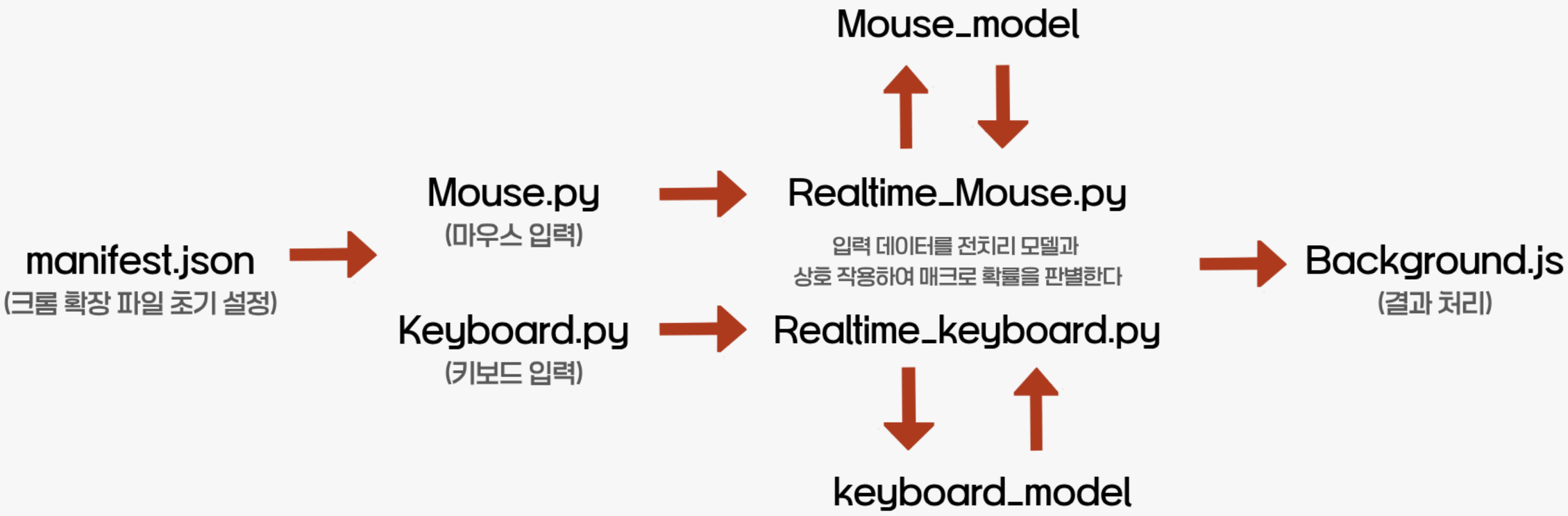
"Bad Bot Sophistication Levels" 인포그래픽
단순 스크립트 매크로부터 인간행동을
흉내내는 고급 봇까지 진화

누구나 매크로를 구할 수 있고, 탐지를 우회하는 봇이
계속 발전하기 때문에 완전히 막는 것은 어렵다

02

만든 프로그램 소개

시스템 구조도


Mouse_model
Mouse.py
(마우스 입력)
Realtime_Mouse.py
manifest.json
(크롬 확장 파일 초기 설정)
입력 데이터를 전치리 모델과
상호 작용하여 매크로 확률을 판별한다
Background.js
(결과 처리)
Keyboard.py
(키보드 입력)
Realtime_keyboard.py
keyboard_model

전체 프로세스

Mouse.py/keyboard.py
입력데이터 추출

```
{ timestamp: 1000, type: "keydown", code: "KeyH" }
{ timestamp: 1085, type: "keyup",   code: "KeyH" }
{ timestamp: 1150, type: "keydown", code: "KeyE" }
```

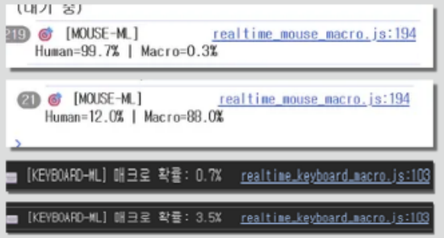
사용자의 실제 마우스, 키보드 입력을 브라우저에서 **실시간**으로 수집

realtime_mouse_*.js
입력 데이터 전처리

```
['p2p_mean', 'p2p_std', 'p2p_min',
'p2p_max', 'dwell_mean', 'dwell_std']
```

Timestamp, x, y, type 등의 데이터를 실제로 사용 가능한 속도 가속도 등의 **데이터**로 바꾼다.

realtime_mouse_*.js + model
전치리 데이터 기반 판별



전처리 데이터를 모델에 입력 후, 사람 행동인지 매크로 행동인지 판별하여 **확률** 제공

Background.js

브라우저 전역에서 들어오는 AI 기반 매크로 탐지 결과를 수집과 기록 하고, 실시간 알림과 리포트 생성으로 사용자에게 경고 및 통계 **대시보드**를 제공

시연 동영상

02-1

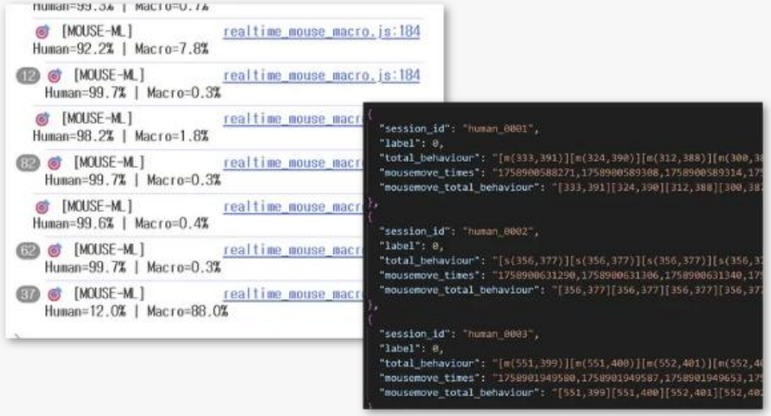
마우스

데이터 설명

timestamp	x	y	type	deltaY
1.76E+12	554	427	move	0
1.76E+12	543	425	move	0
1.76E+12	525	423	move	0
1.76E+12	511	420	move	0
1.76E+12	498	417	move	0
1.76E+12	486	415	move	0
1.76E+12	476	413	move	0
1.76E+12	469	411	move	0
1.76E+12	464	410	move	0

RAW DATA

- 브라우저에서 수집된 실제 마우스 움직임을 기록
- 각 행(row)은 하나의 이벤트를 의미
- 특징으로는 timestamp, x, y, type(move/click/wheel)
- move는 마우스 움직임, click은 마우스 클릭,
wheel은 마우스 스크롤을 의미
- 이 데이터를 기반으로 속도, 가속도 등의
특징(feature) 계산



Preprocessed DATA

- mouse.js에서 수집된 원본 이벤트를
구조화(JSON)한 형태
- 각 session_id 별로 사람 또는 매크로가 수행한 마우스
움직임(m), 클릭(c), 스크롤(s)이 저장
- mousemove_times, total_behaviour, label의 필드로
정리 → 원본(raw) 이벤트를 학습용 Transformer 입력
형태로 변환한 전처리 데이터셋

모델 테스트

머신러닝

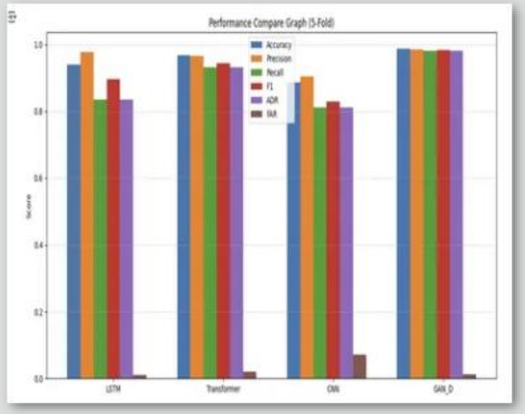
	session_id	mean_dist	std_dist	max_dist	mean_dt	std_dt	max_dt	n_moves	n_clicks	n_scrolls	label
0	hujnm9166tof0hmoeid55a996n	9.614002	17.694452	821.029841	0	0	0	10985	9	195	1
1	ufaj0lbcmqa0l022l9jan5n87l	10.284945	18.590273	896.147309	0	0	0	9468	16	69	1
2	dr09rk5eagjuu87gedvdqmq3gl	5.950566	11.503495	679.018409	0	0	0	8332	11	25	1
3	8h83euesdq85f0shmjm34alpln	9.829202	17.373268	484.134279	0	0	0	11819	14	191	1
4	nvmlnfhs5v6hehsd81e9mf75cn	7.662638	12.321752	280.001786	0	0	0	9763	10	316	1
...	...	...	...	...	...	...	...	...	...	...	...
65	pmb5bepkpun1v95n8rveof5ee0	10.210181	12.807746	699.006438	0	0	0	10139	19	1	0
66	v3tbcnujku5f0bm2mmbjqrr9xg8	10.305012	14.051769	943.002121	0	0	0	12404	19	0	0
67	v6g56j1moj87j05r7n37hjcjqa	10.252564	15.809682	997.0							
68	aj9b410e48gja1a4u0s5e0on3v	10.252127	16.531229	872.0							
69	49rbuft0lmtfjd3qn70np3lk51	8.406641	12.360753	433.1							

70 rows × 11 columns

	session_id	mean_dist	std_dist	max
0	hujnm9166tof0hmoeid55a996n	9.614002	17.694452	821.0
1	ufaj0lbcmqa0l022l9jan5n87l	10.284945	18.590273	896.1
2	dr09rk5eagjuu87gedvdqmq3gl	5.950566	11.503495	679.0
3	8h83euesdq85f0shmjm34alpln	9.829202	17.373268	484.1

정확도: 0.7285714285714285
혼동 행렬:
[[30 5]
[14 21]]

딥러닝



교차검증 최종 결과 요약

- LSTM
  Accuracy: 0.9400
  Precision: 0.9778
  Recall: 0.8345
  F1: 0.8961
  ADR: 0.8345
  FAR: 0.0095

- Transformer
  Accuracy: 0.9667
  Precision: 0.9657
  Recall: 0.9314
  F1: 0.9445
  ADR: 0.9314
  FAR: 0.0213

- CNN
  Accuracy: 0.8867
  Precision: 0.9033
  Recall: 0.8116
  F1: 0.8296
  ADR: 0.8116
  FAR: 0.0718

- GAN_D
  Accuracy: 0.9867
  Precision: 0.9857
  Recall: 0.9818
  F1: 0.9831
  ADR: 0.9818
  FAR: 0.0118

모델 구조

```python
class MouseDataset(Dataset):
    def __init__(self, json_data, max_len=200, screen_w
        self.samples = json_data
        self.max_len = max_len
        self.screen_width = screen_width
        self.screen_height = screen_height

    def __len__(self):
        return len(self.samples)

    def __getitem__(self, idx):
        sample = self.samples[idx]
        label = sample["label"]

        events, coords, times = [], [], []
```

데이터 전처리 및 특징 추출

1. 목적

원본 마우스 이벤트 데이터를 Transformer 입력 맞게 변환하는 전처리 단계
각 세션의 행동 패턴을 정규화, 속도, 가속도로 수치화

2. 방법

total_behaviour 문자열을 파싱
→ (이벤트 타입, 좌표) 추출
좌표를 화면 크기(1920×1080)로 나누어 정규화
인접 프레임 간 속도(speed) 및 가속도(accel) 계산
데이터 길이를 일정하게 맞추기 위해
padding(200 step) 적용

3. 입력 구조

[이벤트코드, norm_x, norm_y, speed, accel]
→ 총 5차원 feature vector × 200 sequence

모델 구조

```
# ========================
class MouseTransformer(nn.Module):
    def __init__(self, input_dim=5, d_model=64, nhead=4
        super(MouseTransformer, self).__init__()
        self.input_fc = nn.Linear(input_dim, d_model)
        self.pos_embedding = nn.Parameter(torch.randn(1
        encoder_layer = nn.TransformerEncoderLayer(d_mo
        self.transformer = nn.TransformerEncoder(encode
        self.fc = nn.Linear(d_model, num_classes)

    def forward(self, x):
        x = self.input_fc(x)
        x = x + self.pos_embedding[:, :x.size(1), :]
        out = self.transformer(x)
        out = out.mean(dim=1)  # 시퀀스 평균 풀링
        return self.fc(out)
```

Transformer Encoder

1. 목적
마우스 행동 시퀀스를 시계열(sequence)로 인식하고,Transformer Encoder로 맥락 기반 패턴 학습을 수행

2. 입력
(batch_size, 200, 5)

3. 출력
[2] → [Human, Macro] 확률 벡터

모델 구조

```
# ========================
def train_model(json_path, epochs=50, batch_size=32, ma
    with open(json_path, "r", encoding="utf-8") as f:
        data = json.load(f)

    dataset = MouseDataset(data, max_len=max_len)

    val_size = int(len(dataset) * val_ratio)
    train_size = len(dataset) - val_size
    train_set, val_set = random_split(dataset, [train_s

    train_loader = DataLoader(train_set, batch_size=bat
    val_loader = DataLoader(val_set, batch_size=batch_s

    device = torch.device("cuda" if torch.cuda.is_avail
    model = MouseTransformer(max_len=max_len).to(device
```

학습 과정

1. 목적

전처리된 데이터셋을 Transformer 모델에 학습시켜
"사람 행동"과 "매크로 행동"의 **차이를 분류**

2. 방법

- Train / Validation Split (8:2)
- DataLoader 로 미니배치(32개씩) 학습
- Loss: CrossEntropyLoss(weight=[1, 5])
  → 매크로 데이터가 적을 경우 가중치 조정

3. 입력 구조

각 Epoch마다:Train Acc, Val Acc, Loss 로그 찍고
학습 완료 후 **trained_model 객체 반환**
최종적으로 ONNX 파일 모델 변환 및 저장

02-2

키보드

데이터 설명

```
{ type: "keydown", code: "KeyH", timestamp: 1000 },
{ type: "keyup",   code: "KeyH", timestamp: 1085 },

{ type: "keydown", code: "KeyE", timestamp: 1150 },
{ type: "keyup",   code: "KeyE", timestamp: 1230 },

{ type: "keydown", code: "KeyL", timestamp: 1290 },
{ type: "keyup",   code: "KeyL", timestamp: 1370 },

{ type: "keydown", code: "KeyL", timestamp: 1450 },
{ type: "keyup",   code: "KeyL", timestamp: 1530 },

{ type: "keydown", code: "KeyO", timestamp: 1600 },
{ type: "keyup",   code: "KeyO", timestamp: 1680 }
```

RAW DATA

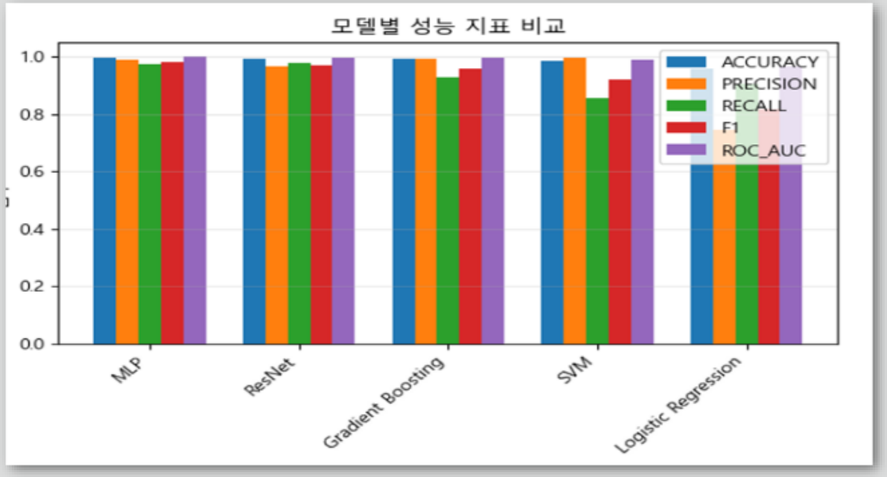
- 브라우저에서 추출한 실제 키보드 입력을 추출
- 입력 데이터는 키 타입, 코드, 타임스탬프
- 이를 토대로 모델 입력데이터를 가공

미리 설정한 윈도우 기반으로 평균 계산

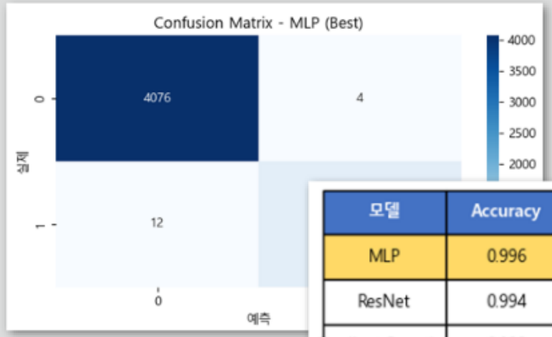
p2p_mean : 평균 입력 간격
p2p_std : 간격 표준편차
p2p_min : 최소 간격
p2p_max : 최대 간격
dwell_mean : 평균 누름 시간
dwell_std : 누름 시간 편차

모델 테스트

모델별 성능 지표



키보드 성능 지표



모델	Accuracy	Precision	Recall	F1
MLP	0.996	0.991	0.973	0.982
ResNet	0.994	0.967	0.978	0.972
Gradient Boosting	0.992	0.993	0.930	0.961
SVM	0.986	0.997	0.858	0.923
Logistic Regression	0.960	0.745	0.908	0.819

모델 구조

```python
class KeyboardDataset(Dataset):
    def __init__(self, json_data, window_size=20):
        self.samples = json_data
        self.window_size = window_size

    def __len__(self):
        return len(self.samples)

    def __getitem__(self, idx):
        sample = self.samples[idx]
        label = sample["label"]

        # 타임스탬프 추출
        timestamps = [event['timestamp'] for event in sample['events']]

        # P2P 간격 계산
        p2p_intervals = []
        for i in range(1, len(timestamps)):
            p2p_intervals.append(timestamps[i] - timestamps[i-1])

        # Dwell Time 계산
        dwell_times = []
        for event in sample['events']:
            if event['type'] == 'keydown':
                # keydown → keyup 찾기
```

데이터 전처리 및 특징 추출

1. 목적

원본 키보드 이벤트 데이터를 MLP 입력 맞게 변환하는 전처리 단계
각 세션의 타이핑 패턴을 6차원 통계 벡터로 수치화

2. 방법

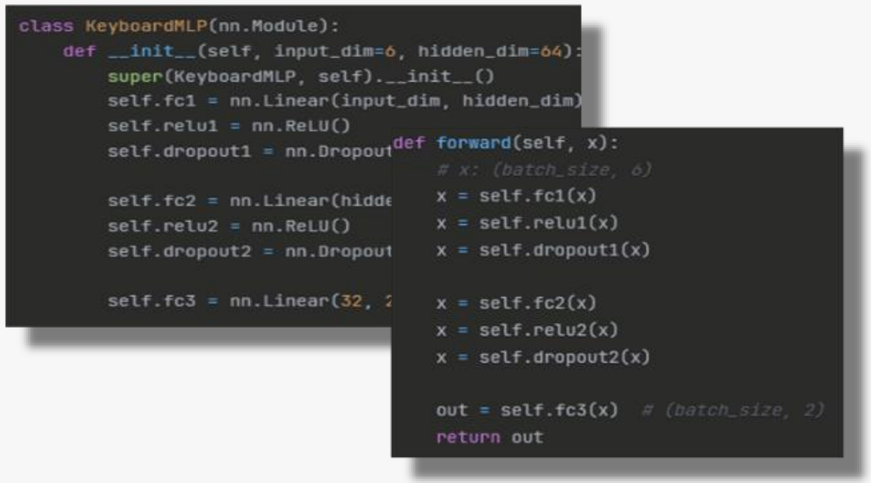
total behaviour 분석을 파싱
→ (이벤트 타입, 타임스탬프) 추출
Press-to-Press 간격 계산 (연속 keydown 차이)
Dwell Time 계산 (keydown → keyup 유지 시간)
통계값 산출: **mean, std, min, max**
데이터 정합성을 위해 **윈도우(20 keys) 적용**

3. 입력 구조

[p2p_mean, p2p_std, p2p_min,
p2p_max, dwell_mean, dwell_std]

→ 총 **6차원 feature vector × 1 sample**

모델 구조



MLP Classifier

1. 목적

키보드 타이핑 통계 특징을 기반으로 사람/매크로 분류
통계 벡터를 MLP로 학습하여 타이핑 패턴의 규칙성 판별

2. 입력

(batch_size, 6)
6차원 통계 벡터: [p2p_mean, p2p_std, p2p_min, p2p_max, dwell_mean, dwell_std]

3. 출력

[2] → [Human, Macro] 확률 벡터
Softmax 적용 후 매크로 확률(0~1) 반환

모델 구조

```python
def train_model(json_path, epochs=100, batch_size=32):
    with open(json_path, encoding="utf-8") as f:
        data = json.load(f)

    dataset = KeyboardDataset(data, window_size=20)

    val_size = int(len(dataset) * 0.15)
    train_size = len(dataset) - val_size
    train_set, val_set = random_split(dataset, [train_size, val_size])

    train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True)
    val_loader = DataLoader(val_set, batch_size=batch_size)

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = KeyboardMLP(input_dim=6).to(device)

    # 클래스 불균형 대응
    criterion = nn.CrossEntropyLoss(weight=torch.tensor([1.0, 5.0]).to(device))
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

학습 과정

1. 목적

전처리된 데이터셋을 MLP 모델에 학습시켜 **차이를 분류**

2. 방법

Train / Validation Split (85:15)
DataLoader로 미니배치(32개씩) 학습
Loss: CrossEntropyLoss(weight=[1, 5])
→ 매크로 데이터가 적을 경우 가중치 조정
Optimizer: Adam (lr=0.001)
Epochs: 100회 반복 학습

3. 입력 구조

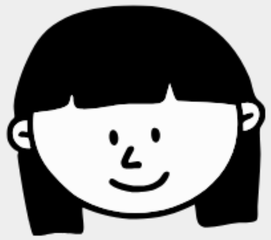
각 Epoch마다 Train Acc, Val Acc, Loss 로그 찍고 학습 완료 후 **trained_model 객체 반환**
최종적으로 ONNX 파일 모델 변환 및 저장
→ 브라우저 환경에서 실시간 추론 가능

03

개발방법

팀원 소개

김송혜



팀장
키보드 부분 확장프로그램 담당

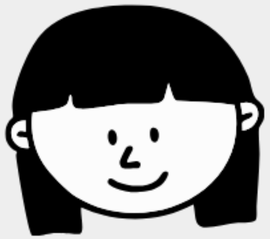
김진섭

키보드 부분 모델 담당

송일환

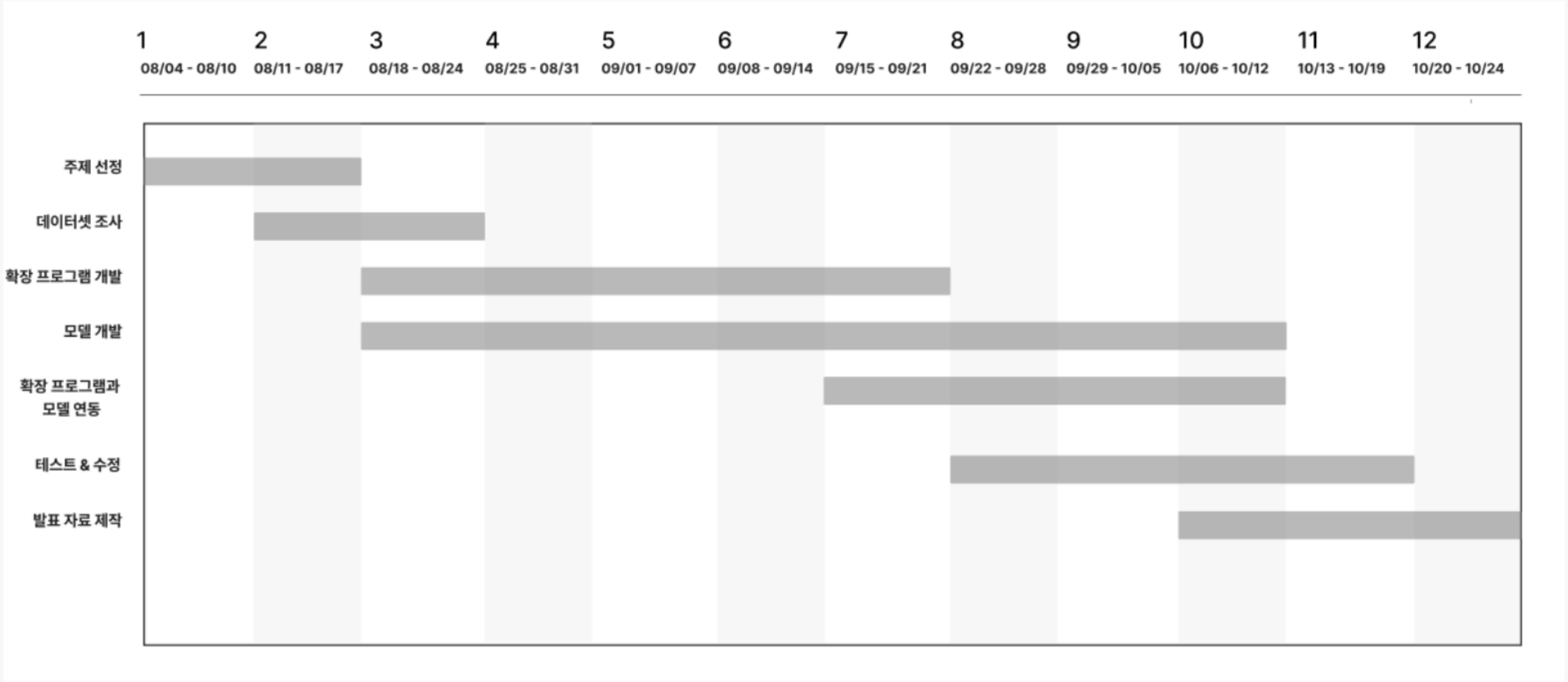
마우스 부분 확장프로그램 담당

최서연



마우스 부분 모델 담당

Time Line


1
08/04 - 08/10
2
08/11 - 08/17
3
08/18 - 08/24
4
08/25 - 08/31
5
09/01 - 09/07
6
09/08 - 09/14
7
09/15 - 09/21
8
09/22 - 09/28
9
09/29 - 10/05
10
10/06 - 10/12
11
10/13 - 10/19
12
10/20 - 10/24
주제 선정
데이터셋 조사
확장 프로그램 개발
모델 개발
확장 프로그램과 모델 연동
테스트 & 수정
발표 자료 제작

04

기대효과

기대 효과

- 자동화 공격(매크로, 봇 등)으로 인한 서비스 불공정 문제 완화
- 사용자 행동 데이터 기반의 지능형 탐지 고도화
- 서비스 신뢰도 및 사용자 만족도 향상
- 확장성 있는 보안 프레임워크로의 발전 가능성

감사합니다.

404 Not Found