사물인터넷융복합논문지 Vol. 11, No. 5, pp. 121-129, 2025 https://doi.org/10.20465/KIOTS.2025.11.5.121

게임 환경에서 실시간 플레이어 분류를 위한 Unity 기반 Transformer 모델 통합 시스템 설계 및 구현

이면재*
백석대학교 컴퓨터공학부 교수

Design and Implementation of a Real-Time Player Classification System in Game Environments Using Unity and Transformer Model

MyounJae Lee*
Professor, Division of Computer Engineering, BaekSeok University

요 약 기존 게임 환경에서의 실시간 플레이어 분류는 주로 규칙 기반 또는 전통적 머신러닝 기법을 활용하며, 서버에서 비실시간으로 처리되어 즉각적인 반영이 어렵다. 이러한 방식은 플레이어의 행동 패턴 변화에 대한 적응력이 낮고, 플레이어의 시퀀스 기반 맥락 이해에 한계가 있어 정밀한 분류가 어렵다는 문제점이 있다. 본 연구는 게임 환경에서 실시간 플레이어 분류를 위해 유니티 엔진에서 Hugging Face 프레임워크 기반 Transformer 모델을 활용하는 방법을 제안한다. 제안된 시스템은 Unity 클라이언트, Flask 서버, Hugging Face Inference API로 구성된 3계층 구조로 구성된다. Unity는 플레이어 데이터를 전처리하여 Flask 서버로 전송하고, Flask 서버는 Transformer 모델에 질의 후 결과를 클라이언트로 반환한다. 이후 이러한 시스템을 구현하는 경우 클라이언트, 개발자, 게임 회사에서 고려할 사항을 기술하였다. 본 연구는 실시간 플레이어 분류를 통해 개인화된 게임 경험, 난이도 조정, 비정상 행위 탐지 등 다양한 게임 내 응용을 가능하게 한다. 특히 Unity 기반 실제 게임 환경에 적용 가능한 AI 시스템 구조를 제시함으로써 산업 현장 활용 가능성을 높이는데 도움을 줄 수 있다.

주제어 : 게임 플레이어, 플레이어 분류, 게임인공지능엔진, 유니티 엔진, Hugging Face

Abstract In traditional game environments, real-time player classification primarily relies on rule-based or conventional machine learning techniques, typically processed on servers in a non-real-time manner, which limits immediate responsiveness. These methods lack adaptability to evolving player behavior and are limited in understanding context based on player action sequences, resulting in low classification accuracy. This study proposes a method for real-time player classification in game environments using a Transformer model based on the Hugging Face framework within the Unity engine. The proposed system adopts a three-tier architecture composed of a Unity client, a Flask server, and the Hugging Face Inference API. Unity preprocesses player data and sends it to the Flask server, which queries the Transformer model and returns the result to the client. The study also outlines considerations for clients, developers, and game companies when implementing such a system. Through real-time player classification, the proposed approach enables various in-game applications such as personalized gameplay, dynamic difficulty adjustment, and abnormal behavior detection. In particular, by presenting an AI system structure applicable to Unity-based game environments, this research contributes to enhancing practical applicability in the gaming industry.

Key Words : Game player, player Classification, Game AI Enginge, Unity Engine, Hugging Face Model

본 논문은 2025학년도 백석대학교 학술연구비 지원을 받아 작성되었음
*교신저자 : 이면재(davidlee@bu.ac.kr)
접수일 2025년 09월 22일 수정일 2025년 10월 15일 심사완료일 2025년 10월 19일

1. 서론

디지털 게임 산업의 성장과 더불어 플레이어 경험(player Experience, UX)을 향상시키고, 서비스 품질을 개선하기 위한 플레이어 행동 분석의 중요성이 점점 더 강조되고 있다[1]. 특히 플레이어 맞춤형 경험(personalized experience)에 대한 수요가 증가함에 따라, 게임 내 행동 데이터를 기반으로 한 정교한 분석과 콘텐츠 최적화는 산업 전반의 경쟁력 확보를 위한 핵심 전략으로 자리매김하고 있다. 게임 플레이어 분류는 플레이어 이탈 예측, 선호 콘텐츠 탐지, 플레이 스타일 분석, 과금 가능성 예측 등과 밀접하게 연관되어 있으며, 이러한 정밀한 분류는 플레이어 유지율 향상과 수익 구조 고도화에도 직접적인 영향을 미친다[2].

기존의 플레이어 분석 방식은 로그 데이터를 기반으로 클릭 수, 세션 시간, 구매 빈도 등 단순 지표를 중심으로 한 통계적 접근법이 주를 이루었다[3]. 이러한 방식은 구현이 간단하고 기본적인 경향 파악에는 유용하지만, 플레이어 행동의 비정형성, 맥락성, 그리고 복잡한 상호작용을 포착하는 데에는 한계가 존재한다. 또한 플레이어 수가 기하급수적으로 증가하고 게임 환경이 복잡해짐에 따라, 정적 분석 방식만으로는 실시간 동적 콘텐츠 설계가 어려워지고 있다[4].

이러한 한계를 극복하기 위해, 최근에는 인공지능(AI) 기술, 특히 머신러닝(Machine Learning)과 딥러닝(Deep Learning)을 활용한 플레이어 행동 분석이 주목받고 있다[5]. 예컨대, 군집분석(Clustering)은 K-Means, DBSCAN 등을 활용하여 유사한 플레이어 그룹을 자동으로 식별하며[6], 시계열 예측(Time Series Forecasting)은 RNN, LSTM 기반 모델을 통해 접속 주기나 과금 패턴 등 시간 기반 행동 예측에 활용된다[7]. 이상 탐지(Anomaly Detection)에서는 핵 사용, 버그 악용 등의 비정상적인 플레이어의 행위를 조기에 식별한다. 추천 시스템(Recommendation System)에서는 과거 플레이 데이터를 기반으로 플레이어 맞춤형 아이템과 콘텐츠를 제안하기도 한다[8][9].

이러한 AI 기반 기법은 복잡하고 비선형적인 플레이어 행동 패턴을 보다 정확하게 포착할 수 있으며, 스트리밍 데이터 기반의 실시간 모니터링 및 반응을 가능하게 하여, 기존의 정적 분석 방식과 차별화되는 높은 정밀도와 확장성을 제공한다[10][11]. 특히 Unity와 같은 실시간 게임 엔진은 방대한 플레이어 데이터를 실시간으로 수집할 수 있는 환경을 제공하며, 이를 외부 분석 시스템과 연동하면 즉각적인 피드백 기반의 게임 콘텐츠 조정도 가능하다.

최근에는 텍스트 로그나 명시적 행동 데이터를 포함한 비정형 정보까지 분석할 수 있는 자연어 처리(Natural Language Processing, NLP) 기반의 AI 기술이 주목받고 있다. 특히 Hugging Face에서 제공하는 Transformer 기반 분류 모델은 Unity 엔진과 연동되어 다양한 텍스트 및 행동 로그 데이터를 분류하는 데 있어 높은 성능을 보이고 있다.

이에 본 논문은 Unity 엔진 기반 게임에서 수집된 플레이어 행동 데이터를 NLP 기반 Transformer 분류 모델을 활용하여 분석하는 방법론을 제시하며, Unity와 외부 Python 서버 간 연동 방식을 통해 실시간 혹은 준실시간 플레이어 분류 시스템 구축 가능성을 검토하고자 한다.

본 논문의 구성은 다음과 같다. 제2장에서는 관련 연구를 살펴보고 3장에서는 Unity 엔진을 이용한 실시간 사용자 분류 시스템에 대한 기술적인 사항들과 구축시 고려사항을 논한다. 그리고 4장에서는 결론을 도출하고 추후 연구 방향을 제시한다.

2. 관련 연구

2.1 플레이어 행동 패턴 분석

게임 환경에서는 플레이어 행동 데이터를 기반으로 다양한 패턴 분석이 이루어진다. 이는 게임 설계 개선, 개인화된 콘텐츠 제공, 플레이어 유지율 향상 등을 위해 필수적인 과정이다.

플레이어 행동 패턴 분석은 플레이어가 게임 내에서 수행하는 다양한 행위를 추적하고 이를 정량적으로 분석함으로써, 플레이어의 성향이나 플레이 스타일을 분류하거나 예측하는 작업이다[1]. 주요 분석 항목은 이동 거리 및 빈도, 전투 횟수 및 성과, NPC와의 상호작용 빈도, 특정 지역 방문 빈도, UI 상호작용 패턴(버튼 클릭, 메뉴 체류 시간 등) 등이며, 이러한 데이터를 기반으로 플레이어가 '전투형', '탐험형', '사회형'과 같은 성향을 갖는지 파악할 수 있다[12]. 지불 정도, 체류 시간, 로깅 횟수도 주요 분석항목에 포함된다. 이러한 분석 과정을 통해서 게임 난이도를 조정하는 동시에 개인화된 콘텐츠를 제공하여 플레이어의 몰입도 및 수익화에 영향을 줄 수 있다[13].

게임 산업에서 사용된 대표적인 플레이어 패턴 분석

방법들중 가장 널리 사용되는 방법은 군집 분석이다.이 기법은 유사한 행동 특성을 보이는 플레이어들을 그룹화함으로써 각 군집의 특성에 맞는 게임 콘텐츠 또는 마케팅 전략을 수립하는 데 활용된다[5]. 예를 들어 League of Legends (Riot Games) 게임에서는 플레이어의 챔피언 선택, 경기 내 행동, 협동 정도 등을 분석해 플레이어 성향을 분류하며, 이를 통해 팀 매칭의 균형을 조정하고 신규 콘텐츠 기획에도 활용한다. 해당 분석에는 K-Means Clustering[14], PCA[15], Autoencoder 등의 기술이 사용된다. [16]에서는 MMORPG인 World of Warcraft의 플레이어 행동 데이터를 바탕으로 K-Means 알고리즘을 적용하여 플레이어들을 탐험형, 사회형, 경쟁형 등의 군집으로 분류하였다. 이러한 분류는 게임 내 상호작용 설계에 실질적인 피드백을 제공한다.

또한, 순차 패턴 분석(Sequential Pattern Mining) 기법도 주목받고 있다. 이 방법은 플레이어의 행동 로그를 시간 순으로 분석하여 특정 행동 패턴이나 루틴을 도출하여 반복적인 게임 실패 구간을 파악해서 이를 개선하여 이탈률 감소에 도움을 주기도 한다. 로그 기반의 순차 분석을 통해 플레이어들이 자주 실패하는 퀘스트 시퀀스를 발견하고, 이를 통해 게임 난이도를 조정한 연구[17]도 있다. 또한 플레이어가 게임을 더 이상 하지 않을 가능성을 예측하는 분석 기법으로 이탈 예측(Churn Prediction)이 있다. 플레이어의 로그인 빈도, 퀘스트 완료율, 커뮤니티 활동 등을 분석해 이탈 가능성이 높은 플레이어를 분류하며, 이를 기반으로 개인화된 이벤트나 보상을 제공하여 재참여를 유도한다.

최근에는 딥러닝 기반의 예측 모델링이 부각되고 있다. 플레이어 이탈률, 결제 가능성 등을 예측하는 데 RNN, LSTM 등의 신경망이 활용된다[17]. 이처럼 게임 플레이어 행동 분석은 전통적인 통계기법에서부터 최신 인공지능 기술까지 다양하게 확장되고 있으며, 게임 설계 및 비즈니스 전략 수립에 사용된다[18]. 핵 사용, 버그 악용 등 일반적이지 않은 행동을 조기에 탐지하는 실시간 이상행동 탐지(Anomaly Detection)에도 딥러닝 기법이 사용되기도 한다[19].

플레이어의 성향에 맞는 콘텐츠(스킨, 퀘스트, 아이템 등)를 추천하는 개인화 콘텐츠 추천(Recommendation)에도 RNN, LSTM등의 딥러닝 방법이 활용된다[20]. Steam 플랫폼에서는 플레이어의 게임 이력, 친구 목록, 플레이 시간 등을 기반으로 플레이어에게 콘텐츠를 추천하여 플레이어 참여도를 높이고 매출을 증대시키기도 하였다[21].

2.2 Unity 엔진 기반 게임에서의 플레이어 패턴 분석

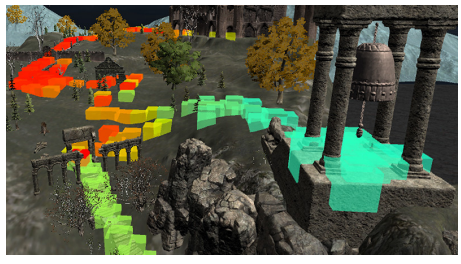
Unity 엔진은 게임 개발에 널리 활용되는 통합 개발 환경으로, 플레이어 로그 수집과 분석을 위한 다양한 기능을 제공한다. 특히 C# 스크립트를 통해 플레이어 행동 데이터를 실시간으로 수집하고, 이를 기반으로 한 다양한 패턴 분석 기법을 적용함으로써 게임의 품질 및 플레이어 경험을 향상시킨다. 본 절에서는 Unity 환경에서 대표적으로 활용되는 플레이어 패턴 분석 방법들을 살펴본다. Unity 엔진은 게임 개발에 널리 활용되는 통합 개발 환경으로, 플레이어 로그 수집과 분석을 위한 다양한 기능을 제공한다[22].

2.2.1 이벤트 기반 로그 수집 및 분석 (Event-driven Logging)

이벤트 로그 기반 분석은 특정 기능 사용률, UI 전환 경로, 게임 종료 시점 등을 파악하는 데 효과적이며, UX 최적화에 활용된다. Unity 엔진에서는 인터랙션, 버튼 클릭, 이동 경로, 퀘스트 수행 등과 플레이어 이벤트들을 Analytics. CustomEvent() API를 사용하여 실시간으로 기록할 수 있다[22]. 이러한 이벤트 로그는 Google Firebase, Unity Analytics 또는 외부 데이터베이스(DB)로 전송되고 게임 기획자는 이 정보들을 분석하여 게임 유지보수에 반영한다.

2.2.2 히트맵 분석(Heatmap Visualization)

Unity에서는 플레이어 위치 데이터를 기반으로 히트맵을 생성하여 플레이어의 사망 위치, 자주 방문한 지역, 아이템 습득 위치 등을 시각적으로 표현하는 기능을 제공한다[23]. 이 정보는 레벨 디자인 평가시 활용된다. 히트맵은 자체 구현하거나 Unity Asset Store의 플러그인(예: HeatmapTool)을 활용해 손쉽게 시각화할 수 있다. [Fig. 1]은 Unity 히트맵의 예이다[31].



[Fig. 1] Unity Heatmap

2.2.3 시퀀스 분석 및 행동 흐름 모델링

Unity 엔진에서는 게임 내 플레이어 행동을 시간 순서대로 분석하는 시퀀스 분석도 수행할 수 있다[24]. 로그 데이터를 JSON 혹은 CSV 형태로 저장한 후, 외부 도구(R이나 Python 등)를 통해 행동 흐름을 모델링할 수 있다. 예를 들어, 튜토리얼 → 전투 → 아이템 구매 → 종료와 같은 순서를 분석하여 이탈 원인을 진단하거나, 행동 루틴을 기반으로 게임 밸런싱을 조정할 수 있다. [Fig. 2]는 이 과정을 보여준다. Unity에서 플레이어의 행동을 Log로 저장하고([Fig. 2](a) 이 로그 파일을 플레이어별로 목록화한다([Fig. 2](b)). 이후에 이후에 튜토리얼 이후에 곧바로 나왔는지 배틀을 치르고 게임이 종료되었는지를 분석하여([Fig. 2](c)) 게임 밸런싱을 조정한다.

```
[
  {"user_id": "user123", "timestamp": "2025-09-20T12:00:00", "action": "Start Tutorial"},
  {"user_id": "user123", "timestamp": "2025-09-20T12:05:00", "action": "Enter Battle"},
  {"user_id": "user123", "timestamp": "2025-09-20T12:10:00", "action": "Purchase Item"},
  {"user_id": "user123", "timestamp": "2025-09-20T12:12:00", "action": "Exit Game"},

  {"user_id": "user456", "timestamp": "2025-09-20T13:00:00", "action": "Start Tutorial"},
  {"user_id": "user456", "timestamp": "2025-09-20T13:03:00", "action": "Exit Game"},
```

(a) Log from Unity

```
# Load JSON data
with open("user_log.json", "r", encoding='utf-8') as f:
    data = json.load(f)

# Convert to DataFrame
df = pd.DataFrame(data)
df['timestamp'] = pd.to_datetime(df['timestamp'])
df = df.sort_values(by=['user_id', 'timestamp'])

# Extract sequences by user
sequences = df.groupby('user_id')['action'].apply(list)
print(sequences)
```

(b) Sequence Analysis

```
user_id
user123    [Start Tutorial, Enter Battle, Purchase Item, Exit Game]
user456    [Start Tutorial, Exit Game]
```

(c) Detailed analysis

[Fig. 2] Sequence Analysis using Unity.

2.2.4 머신러닝 기반 플레이어 분류 및 예측

Unity에서는 ML-Agents Toolkit을 사용하여 머신러닝 모델을 훈련하고, 플레이어 데이터를 분석하는 기능을 제공한다[25]. 이를 통해 특정 행동 패턴을 보이는 플레이어를 자동으로 분류하거나, 구매 가능성 및 이탈 위험 등을 예측하는 시스템을 구현할 수 있다. 또한, 게임 내 AI와 플레이어 행동 예측을 연계하여 더욱 정교한 게임 디자인이 가능하도록 도와준다. 이와같이 Unity 엔진에서는 다양한 분석 기법들을 유기적으로 활용할 수 있으며, 실시간으로 플레이어 피드백을 받을 수 있다. 이 피드백은 게임 설계 및 운영 전략에 반영하는 데 중요하다[26]. Unity ML-Agents Toolkit은 Unity 내부에서 강화학습 환경을 직접 구성할 수 있도록 지원하는 툴 킷으로, 게임 내 NPC 또는 적 AI의 행동을 플레이어의 패턴에 따라 적응시키는 데 사용된다. ML-Agents는 Python API를 통해 외부의 학습 모델과 통신하며, Unity 내에서 환경을 시뮬레이션함으로써 반복적인 학습과 보상을 수행할 수 있다[29].

2.3 Unity 엔진 기반 인공지능엔진

최근 머신러닝 및 자연어 처리 기술이 발달됨에 따라 Unity 엔진에서는 플레이어 행동을 보다 정교하게 해석하고 예측할 수 있는 다양한 인공지능 엔진과 연계하는 시도가 활발히 진행되고 있다. 본 절에서는 Unity 기반 게임에서 플레이어 분석을 목적으로 사용되는 주요 인공지능 엔진의 특징과 적용 방식을 고찰한다.

우선, 자연어 처리 분야에서 널리 활용되는 Hugging Face의 Transformers 라이브러리는 플레이어 행동 데이터를 문장 형태로 요약한 후, 이를 분류하거나 감정을 분석하는 데 효과적으로 활용된다[27]. 예를 들어, “플레이어가 20분 동안 NPC와 대화하였고, 전투에는 참여하지 않았다”는 입력은 사전학습된 분류 모델(BERT, RoBERTa 등)에 의해 ‘사회적 상호작용 성향’과 같은 결과로 해석될 수 있다. 이와 같은 자연어 기반 분석은 Python 환경에서 수행되며, Unity와의 연동은 일반적으로 RESTful API를 통해 실현된다.

또한, TensorFlow 및 PyTorch와 같은 범용 딥러닝 프레임워크와 연계하는 경우 게임 내 행동 로그 데이터를 기반으로 플레이어 군집화, 행동 예측, 리스크 평가 등 복잡한 분석을 수행할수 있다[28]. 이 프레임워크들에서는 시계열 예측, 강화학습, 클러스터링 등의 모델을 학습시켜서 플레이어의 향후 행동을 예측하거나 특정 플레이어 집단의 특성을 도출한다.

이러한 플레이어에 대한 분석 결과는 실시간으로 콘텐츠 진행에 변화를 주거나, 난이도를 조정하거나 추천 시스템 등에 활용될 수 있다.

이외에도 Unity 엔진과 OpenAI, Google Cloud Vertex AI[30], AWS SageMaker 등 클라우드 기반 AI 엔진과 연동하는 방식도 존재한다.

그러나, 오픈모델을 공유하고 재사용하는 과정에서 발생할 수 있는 보안 취약성을 실험적으로 탐색한 연구도 있다. 특히 [32]연구에서는 Hugging Face 모델 허브상에 존재하는 모델들에 대해 악용 가능성을 분석하였다.

Hugging Face Model에 대한 개발 사례로는 iPhone 앱인 HuggingSnap을 개발해, 실제 기기상에서 비디오 이해까지 가능한 모델을 시도한 연구[33]가 있다. 모델 크기뿐 아니라 지연(latency), 전력 소비 등 실사용 관점에서의 최적화를 고려하였다.

종합하면, Unity 엔진에서 플레이어 분석을 위해 사용되는 인공지능 엔진은 분석 목적과 데이터 형태에 따라 선택되며, 대체로 Unity와 외부 AI 서버 간 통신을 기반으로 하는 클라이언트-서버 구조를 취한다.

이러한 구조는 실시간 반응성과 유연한 확장성을 동시에 확보할 수 있다는 점에서 점차 보편화되고 있다.

3. 구현

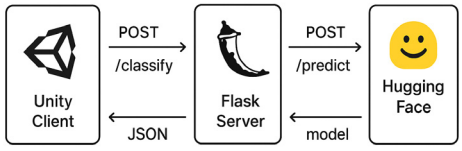
본 장에서는 실시간 플레이어 분류를 위한 하드웨어 구성과 구현 방법 그리고 제안된 시스템 구현시 고려사항을 개발자 관점, 클라이언트 관점, 개발사 관점으로 기술한다.

3.1 하드웨어 구성

Unity 클라이언트, Python 기반의 Flask 프레임워크를 갖는 서버로 구성된다. 이를 구현하기 위해서는 클라이언트에서는 유니티 엔진의 최소 사양을 따라야 한다. Python은 3.8~3.10, Flask 2.2+, Transformers 4.33+이상, Unity엔진은 2021.3 LTS 이상으로 구현한다.

Unity 클라이언트에서는 플레이어의 행동 또는 입력 데이터를 수집하고 Flask 서버로부터 전송된 결과값을 받아 플레이어에게 보여준다. Python 기반의 Flask 서버에서는 Hugging Face Transformers 모델을 로드하여 Unity 클라이언트와 Hugging Face Transform 모델 사이에서 중간자 역할을 수행한다. Flask는 Python으로 작성된 경량 웹 프레임워크로, 간단하고 유연한 웹 애플리케이션을 개발하는 데 적합하여 초보자도 쉽게 사용할 수 있으며, 확장성이 뛰어나 복잡한 프로젝트에도 활용 가능하다.

Unity 클라이언트에서는 Flask 서버에 데이터를 전달한다. 이후 Flask 서버에서는 Hugging Face Transformer 모델에 API를 이용하여 데이터를 전달한다. 데이터를 전달받은 Hugging Face Transformer 모델에서는 플레이어의 패턴을 실제 분석한다. 감정, 성향, 행동 패턴과 같은 플레이어 입력을 분류하여 REST API 또는 WebSocket으로 Flask 서버를 통해 Unity 클라이언트에 분석된 결과 값을 보여준다. Hugging Face는 다양한 자연어 처리 모델을 API 혹은 파이프라인 형태로 제공하며, 플레이어의 반응 감정 측정을 측정하거나 문장을 분류하여 행동 요약 데이터를 바탕으로 성향을 분류하거나 플레이어의 행동을 요약 또는 추천 콘텐츠를 생성하는데 활용된다. 예를 들어, nlptown/bert-base-multilingual-uncased- sentiment 모델은 플레이어 행동을 정성적으로 분석하여 '긍정적'인지 '부정적'인 성향을 갖는 플레이어인지 분.류한다. [Fig. 3]은 이를 위한 시스템 아키텍처를 보여준다.



[Fig. 3] Architecture for player Classfication

3.2 구현 방법

구현한 서버 컴포넌트는 Python 기반의 Flask 프레임워크를 사용하여 제작되었으며, 이 프레임워크는 유니티 게임 클라이언트와 Hugging Face API 간의 중계 게이트웨이 역할을 수행한다. 제안된 구조는 Hugging Face API에 대한 직접 접근을 추상화하고, API 인증 토큰을 안전하게 보호하며, 모델 출력 결과를 클라이언트로 전달하기 전에 전·후처리를 수행할 수 있도록 설계되었다.

Unity 클라이언트에서는 게임 내에서 수집된 플레이어 입력, 행동 로그, 대화 텍스트 등의 데이터를 전처리한 후, POST /classify 메서드를 통해 Flask 프레임워크가 탑재된 Python 서버로 전송한다. 전송되는 메시지는 JSON 형식으로 { "text": "〈플레이어 입력 데이터〉" } 구조를 가지며, HTTP 프로토콜을 기반으로 통신한다. 〈Table 1〉은 플레이어의 행동 로그의 예를 보여준다.

〈Table 1〉 Player Action Log

Time	Action Description
10:01:12	Player1 moved to (12, 8)
10:01:18	Player1 interacted with NPC 'Merchant'
10:01:22	Player1 opened chest 'TreasureBox2'
10:01:27	Player1 used item 'HealthPotion'
10:01:31	Player1 attacked Enemy 'Goblin' with 'Bow'
10:01:35	Player1 moved to (14, 9)
10:01:39	Player1 opened chest 'TreasureBox2'
10:01:42	Player1 used item 'ManaPotion'
10:01:45	Player1 attacked Enemy 'Goblin' with 'Bow'
10:01:50	Player1 interacted with NPC 'Blacksmith'

〈Table 2〉는 Unity 클라이언트의 의사 코드를 보여준다. 클라이언트에서 저장한 로그 파일을 라인 단위로 읽어서 action list에 추가하고 이 리스트들을 Flask 서버가 탑재된 Python 서버에 전송한다. SendLogs메소드는 완성된 로그 목록을 하나의 문자열로 병합하고 병합된 문자열을 JSON 형식으로 구성하여 Flask가 탑재된 서버에 전송한다. [Fig. 4]는 Flask서버에서 연동되는 JSON 데이터의 예를 보여준다.

〈Table 2〉 Pseudo Code for the Unity Client

```
public class ActionLogSender : MonoBehaviour{
  string serverUrl = "http://localhost:xxxx/classify";
  List<string> actionLogs = new List<string>();
  void Start()    {
  //Begin asynchronous process to load logs and send to server
   StartCoroutine(LoadLogsAndSend());
   }
   IEnumerator LoadLogsAndSend()    {
     For each line in the log file:
      Trim the line and add it to the action list
     SendLogS();//Send the collected logs to the server
    }
   IEnumerator SendLogs() {
    string json = JsonUtility.ToJson(new Message {
    text = string.Join("\n", actionLogs) });
    Create HTTP POST request to serverUrl
     Set request body to JSON-encoded log data
     Set request header: Content-Type = "application/json"
     Send request and wait for response
    }
    [System.Serializable]
    public class Message { public string text; }
}
```

```
{"time": "10:01:12", "action": "Player1 moved to (12, 8)"},
{"time": "10:01:18", "action": "Player1 interacted with NPC 'Mer
```

[Fig. 4] Example of JSON Data

〈Table 3〉은 Flask 서버의 의사 코드를 보여준다. Flask 서버는 수신한 데이타를 파싱하고 파싱된 데이터를 Hugging Face Inference API의 입력 스키마에 맞게 변환한다. 이를 위해 Python requests.post() 메서드를 사용하며, 헤더에는 Authorization: Bearer 〈HF_TOKEN〉 형식의 인증 토큰을 포함한다. 전송되는 메시지는 { "inputs": "〈텍스트 데이터〉" } 형태를 따른다. 인증 토큰은 보안 유지를 위해 환경 변수에서 불러오도록 구성되어 있으며, 클라이언트 코드나 버전 관리 저장소에 직접 노출되지 않는다.

서버에서는 /classify라는 단일 RESTful 엔드포인트를 제공하며, 클라이언트로부터 HTTP POST 요청을 수신한다. 그리고 사전 학습된 Hugging Face 모델을 로드한다(단계 3). 이 연구에서는 ert-base-uncased fine-tuned on custom game behavior dataset를 모델로 사용한다. 유니티 클라이언트로부터 받은 데이터를 data에 저장하고(단계 4) logs라는 키를 갖는 값만을 추출하여 logs에 저장하고 플레이어의 행동을 하나의 문자열로 만든다. action_sequence에는 Player1 moved to (12, 8)', 'Player1 interacted with NPC"와 같은 플레이어의 행동이 순차적으로 저장된다. 이후에 classifier 메소드를 사용하여 Hugging Face 모델에게 action sequence를 전달하면(단계 5) Hugging Face 모델에서는 플레이어의 성향을 결과 값으로 제공한다. 이 결과값은 prediction에 저장한다. 이때 prediction 값은 저장된 값은 플레이어의 타입과 이 타입에 대한 신뢰도가 점수 형태로 제공된다.

〈Table 3〉 Pseudo code for the Flask server

```
1. INITIALIZE web application using Flask
2.  LOAD configuration parameters:
(HF_TOKEN, MODEL_URL, HTTP headers)
"https://api-inference.huggingface.co/models/your-model-id"
headers = {"Authorization": "Bearer YOUR_HUGGINGFACE_TOKEN"}
3.Load a pre-trained Hugging Face sequence classifier for custom
model)
classifier=pipeline("text-classification",        model="bert-base-uncased
fine-tuned on custom game behavior dataset ");
4. When a request is received (data = request.get_json());
    logs = data['logs'];
    action_sequence = " → ".join([entry['action'] for entry in logs])
5. prediction = classifier(action_sequence));
Forward the text to a pretrained classification model via Hugging Face
API
6. Receive the predicted player behavior type and confidence score
7. Return the result as a JSON response to the client
eturn
 jsonify({
          "predicted_behavior": prediction[0]['label'],
          "confidence": prediction[0]['score']
    }
```

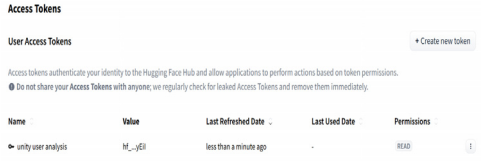
Hugging Face Model에서는 지정된 Transformer 모델(예: BERT, DistilBERT 등)을 이용하여 입력 데이터를 분류하고, 예측 결과를 JSON 형식으로 반환한다. 플레이어 1의 경우 "Player1 moved to (12, 8) → Player1 interacted with NPC 'Merchant' → Player1 opened chest ..."와 같은 형태로 action sequence에 저장된다. 이후에 (단계 5)의 과정이 수행된다. 그리고 최종 결과는 [Fig. 5]와 같은 형태로 유니티 클라이언트에게 전송된다. 현재 이 예에서 Player1은 공격적인 탐험가 스타일로 0.92정도의 확률 즉 정확도를 갖는다는 의미이다.

```
{
  "predicted_behavior": "Aggressive Explorer",
  "confidence": 0.92
}
```

[Fig. 5] Result Example

Flask 서버는 수신한 예측 결과를 그대로 Unity 클라이언트에 전달하거나 후처리 로직을 적용하여 반환한다. 반환되는 응답 메시지는 JSON 구조를 유지하며, 게임 클라이언트는 해당 결과를 기반으로 난이도 조정, 보상 체계 변경, NPC 대사 선택 등 맞춤형 게임 경험을 제공한다.

이러한 시스템을 구현하기 위해서 4단계의 준비작업이 필요하다. 첫번째 단계는 Hugging Face 모델 선택과 모드, 인증 토큰을 생성하는 단계이다. 이를 위해 회원가입 및 로그인이 필요하며 로그인을 수행한 후 인공지능 모델을 선택한다. Hugging Face 모델 선택 또는 업로드 2가지 방법이 있다. 공개모델로는 GPT2, BERT, Whisper, Stable Diffusion 등을 사용할 수 있으며 자신이 학습시킨 모델을 직접 사용할 수 있다. 이후 선택된 모델과 통신할 수 있는 토큰을 생성한다. [Fig. 6]은 토큰이 생성된 결과를 보여준다. 이 토큰은 〈Table 3〉에서 Flask 프레임워크가 Hugging Face Model과 통신하기 위한 헤더 정보로 사용된다.



[Fig. 6] Token

두 번째 단계는 flask 서버를 구축하는 것이다. 〈Table 3〉과 같은 소스를 Flask 서버의 시작점(Entry Point)인 app.py에 저장한다. 운영 환경에서는 SSL/TLS 암호화를 적용하고, WSGI 호환 서버(예: Gunicorn)를 통해 배포하여 안전하고 안정적인 클라이언트–서버 통신을 보장할 수 있다.

이와 같은 중계 서버 구조는 MODEL_URL만 변경함으로써 손쉽게 Transformer 모델을 교체할 수 있으며, 추가적인 데이터 전·후처리 모듈을 유연하게 통합할 수 있다. 따라서 게임 플레이어 맞춤형 콘텐츠를 제공하는 적응형 게임 퍼스널라이제이션 시스템에 특히 적합하다.

3.3 고려사항

유니티 엔진에서 실시간으로 Hugging Face 프레임워크의 Transformer 모델을 활용하여 플레이어 분류를 수행할 경우, 다음과 같은 고려사항이 중요하다.

본 연구에서 제안하는 실시간 플레이어 분류 시스템은 게임 환경에서 실질적으로 사용되기 위해, 클라이언트 이용자에게 안정적인 서비스를 제공함과 동시에, 개발자 및 게임 서비스 제공자의 입장에서 시스템 유지보수, 확장성, 보안성 등 다양한 측면을 고려해야 한다. 이에 따라 다음과 같은 세 가지 관점에서의 주요 고려사항을 제안한다.

3.3.1 클라이언트(이용자) 관점

첫째, 실시간 분류 시스템은 플레이 도중 플레이어의 행동을 기반으로 즉각적인 반응(예: 난이도 조절, 보상 시스템 조정 등)을 유도하므로, 추론 결과의 지연(latency) 은 플레이어 경험에 직접적인 영향을 미친다. 이를 위해 클라이언트측에서는 서버 응답 대기 중 기본 정책을 적용하거나 로딩 UI를 활용하여, 결과 도출 전 단계에서의 UX 공백을 최소화해야 한다.

둘째, 클라이언트는 플레이어의 개인정보 또는 행동 로그를 서버로 전송하게 되므로, 데이터 보안 및 개인 정보 보호에 대한 명확한 안내와 동의 절차가 필요하다. 이는 이용자의 신뢰 확보와 동시에, 데이터 수집의 법적 정당성을 확보하기 위한 필수 요소이다.

셋째, 분류 결과가 게임 내 중요한 의사결정에 활용되는 경우(예: 플레이 스타일 기반 보상 차등 지급), 잘못된 분류로 인해 플레이어가 불공정한 게임 경험을 겪지 않도록, 결과 반영 방식은 점진적이고 보조적인 수준에서 적용되어야 한다.

3.3.2 개발자(시스템 구축자) 관점

첫째, 추론 지연 최소화를 위해 경량화된 Transformer 모델을 채택하거나, 비동기 처리 구조(async 또는 큐 기반 메시징)를 도입하여 서버 처리 병목을 방지해야 한다. 필요 시 Flask 대신 FastAPI, gRPC 등의 프레임워크 도입도 고려할 수 있다.

둘째, API 인증을 위해 JWT등과 같은 보안 인증 체계를 도입하고, 클라이언트의 인증 정보는 하드코딩이 아닌 안전한 환경변수 또는 암호화 방식으로 관리해야 한다. 플레이어 로그 전송 시에는 HTTPS 기반의 암호화된 통신을 사용하고, 민감 정보에 대해서는 별도의 암호화 알고리즘을 적용해야 한다.

셋째, 다양한 문화권의 플레이어 행동 로그를 정확히 반영하기 위해, 도메인 특화 데이터셋 수집과 모델 파인튜닝(fine-tuning) 프로세스를 병행해야 한다. 또한, 추후에 새로운 데이터가 쌓일 경우를 대비해 자동 학습 파이프라인(AutoML or Human-in-the-loop) 의 설계도 필요하다.

넷째, 서버 장애, 외부 API 실패 등 예외 상황을 고려한 Fallback 로직, 타임아웃 처리, 캐싱 시스템 등을 함께 구축해야 하며, 예측 실패 시에도 플레이어 경험이 중단되지 않도록 대응 로직을 마련해야 한다.

다섯째, 분류 결과는 예측 신뢰도(score)와 함께 반환되어야 하며, 결과 신뢰도가 낮은 경우 자동 무시하거나 기본값을 적용하는 로직이 필요하다. 이와 함께, 분류 결과 로그를 기록하고 주기적으로 검증할 수 있는 분석 및 테스트 환경 구축이 중요하다.

3.3.3 게임 회사(서비스 제공자) 관점

첫째, AI 시스템이 게임 운영에 적용되는 만큼, 분류 모델이 플레이어 경험에 미치는 영향을 지속적으로 모니터링하고, 특정 유형의 플레이어가 불이익을 받지 않도록 편향 여부를 판단하는 검증 체계를 갖추어야 한다.

둘째, 수집되는 플레이어 데이터는 비식별화하여 저장하고, 이용자의 동의를 받은 경우에만 모델 학습용으로 활용 가능하도록 해야 한다. 이를 위해 게임사는 데이터 관리 정책을 수립하고, 법적 기준(예: 개인정보보호법)을 충족시켜야 한다.

셋째, AI 모델은 게임 업데이트 주기, 콘텐츠 변화에 따라 교체 또는 개선이 필요할 수 있으므로, 모델 교체 시 기존 시스템에 영향을 주지 않도록 버전 관리 및 Hot-Swap이 가능한 구조로 설계되어야 한다.

넷째, AI 기술의 적용이 이용자에게 신뢰를 줄 수 있도록, AI 활용 목적, 방식, 보호 정책에 대한 투명한 고지가 중요하다. 이를 통해 게임사는 기술적 신뢰뿐 아니라 브랜드 신뢰도 또한 확보할 수 있다.

이와 같은 다각적인 고려사항을 기반으로 본 시스템은 기술적 완성도뿐 아니라 플레이어 친화성, 운영 안정성, 윤리적 책임성까지 포괄하는 현실적인 게임 AI 시스템 모델로 자리매김할 수 있을 것이다.

4. 결론 및 추후 연구방향

게임 환경 플레이어의 행동 패턴과 선호도를 기반으로 한 플레이어 성향 분류는 플레이어의 몰입에 도움을 주며 맞춤형 콘텐츠를 제공하는데 도움을 준다. 정확한 플레이어의 성향이나 타입 분류는 게임 난이도 조정, 보상 체계 설계, 사회적 상호작용 구조 최적화 등 핵심 게임 메커니즘의 개인화에 기여한다.

따라서 플레이어 분류는 장기적인 이용자 유지와 수익성 증대에 직결되는 필수적인 연구 및 구현 요소이다. 본 연구에서는 다양한 플레이어 분류 방법중 유니티 엔진에서 Hugging Face 프레임워크의 Transformer모델을 활용한 플레이어 분류 방법을 제시하고 구현한 예를 보여준다.

추후에는 제안된 시스템을 구현하여 FPS나 실시간 응답지연과 같은 평가요소로 테스트를 진행할 예정이다.

REFERENCES

[1] Q. Yang and Y. Gong and G. Liu, “User Behavior Analysis and Clustering in a MMO Mobile Game: Insights and Recommendations”, arXiv, Jul. 2024. [Online]. Available: https://arxiv.org/abs/2407.11772

[2] D. Melhart, A. Azadvar, A. Canossa, A. Liapis and G. N. Yannakakis, “Your Gameplay Says It All: Modelling Motivation in Tom Clancy’s The Division”, 2019 IEEE Conference on Games (CoG), pp.1-8, Sept. 2019.

[3] L. Yang, et al., “Large-scale Personalized Video Game Recommendation via Social-aware Contextualized Graph Neural Network”, arXiv, Feb. 2022. [Online]. Available: https://arxiv.org/abs/2202.03392.

[4] A. Saas and A. Guitar and Á. Periáñez, “Discovering Playing Patterns: Time Series Clustering of Free-To-Play Game Data” arXiv, Oct. 2017. [Online]. Available: https://arxiv.org/abs/1710.02268

[5] H. Sifa and A. Drachen and C. Bauckhage, “Large-Scale Cross-Game Player Behavior Analysis on Steam”, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, No.11, No.1, pp.198-204, 2021.

[6] M. Ester and H.-P. Kriegel and J. Sander and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), pp. 226–231, 1996.

[7] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[8] J. MacQueen, “Some Methods for Classification and Analysis of Multivariate Observations,” Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281-297, 1967.

[9] Y. Koren and R. Bell and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems”, Computer, vol. 42, no. 8, pp. 30–37, 2009.

[10] Y. LeCun and Y. Bengi and G. Hinton, “Deep Learning” Nature, vol. 521, pp. 436–444, 2015.

[11] D. M. Blei and A. Y. Ng and M. I. Jordan, “Latent

Dirichlet Allocation", Journal of Machine Learning Research, vol.3, pp. 993-1022, 2003.

[12] Richard Bartle, "HEARTS, CLUBS, DIAMONDS, SPADES: PLAYERS WHO SUIT MUDS", Journal of MUD research, No.1, pp.19-36, 1996.

[13] Eva Ascarza and Oded Netzer and Julian Runge, "Personalized game design for improved user retention and monetization in freemium games", International Journal of Research in Marketing, pp. 1-54, 2024.

[14] Ong, H. Y and Deolalikar and S., & Pe "Player Behavior and Optimal Team Composition for Online Multiplayer Games", arXiv preprint arXiv:1503.02230, 2015.

[15] Jack J, "Content-Based Modelling with PCA For Champion Recommendation (League of Legends)", Towards Data Science, 2020. (https://www.itero.gg/articles/champ-recommend).

[16] Drachen, A. and Canossa, A. and Yannakaki and G. N, "Player modeling using self-organization in Tomb Raider: Underworld", IEEE Symposium on Computational Intelligence and Games, 1-8, 2009.

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", Neural Computation, vol.9, no.8, pp.1735-1780, 1997.

[18] R. Drachen and C. Thurau and J. Sifa and C. Bauckhage, "A Comparison of Methods for Player Clustering via Behavioral Telemetry",Proceedings of the FDG, pp. 245-252, 2013.

[19] F. T. Liu and K. M. Ting and Z.-H. Zhou, "Isolation Forest", Proceedings of the 2008 IEEE International Conference on Data Mining, pp. 413-422, 2008.

[20] J. Bobadilla and F. Ortega and A. Hernando and A. Gutiérrez, "Recommender Systems Survey",Knowledge-Based Systems, vol.46, pp. 109-132, 2013.

[21] Stefanidis and Konstantinos and Anastasios Papadopoulos and Yannis Theodoridis. "A Hybrid Recommender System for Steam Games", Proceedings of the 11th International Symposium on Image and Signal Processing and Analysis (ISPA), 2019, pp.191-196. IEEE.

[22] Unity Technologies, Unity Manual: Introduction to Unity", Unity Documentation, 2025.

[23] https://www.reddit.com/r/VALORANT/comments/1fgpm4j/i_made_a_site_that_creates_player_heatmaps_and/?tl=ko.

[24] Guenter Wallner, "sequential Analysis of Player Behavior", Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play, pp.349-358, 2015.(https://doi.org/10.1145/2793107.2793112)

[25] https://elice.io/ko/newsroom/unity-ml-agents

[26] https://unity-technologies.github.io/ml-agents/ML-Agents-Toolkit-Documentation/?utm_source=chatgpt.com.

[27] https://unity.com/blog/games/hugging-face-ai-models-and-more-sentis-updates.

[28] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems", arXiv preprint arXiv:1603.04467, 2016.

[29] Unity Technologies, "ML-Agents Toolkit Documentation" ,Unity, 2025.

[30] Google Cloud, "Vertex AI Overview," Google Cloud Documentation, 2025.

[31] https://docs.unity3d.com/2020.1/Documentation/Manual/UnityAnalyticsOverview.html.

[32] Casey, B and Santos, J. C. S., and Mirakhorli, M, "A Large-Scale Exploit Instrumentation Study of AI/ML Supply Chain Attacks in Hugging Face Models", arXiv-Cryptography and Security, 2024.(arXiv. https://doi.org/10.48550/ARXIV.2410.04490).

[33] merve, Sergio Panieogo, Aritra Roy Gosthipaty, Petro Cuenca, Andres Marafioti, "Vision Language Models (Better, Faster, Stronger)", 2025. (https://huggingface.co/blog/vlms-2025?utm_source).

이 면 재(MyounJae Lee) [종신회원]

■ 2009년 3월 ~ 현재 : 백석대학교 컴퓨터공학부 교수

〈관심분야〉
사물인터넷, 게임, MPEG