Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.1, Feb. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.1.93

자기 유사도를 이용한 MMORPG 게임봇 탐지 시스템

이 은 조,[1†] 조 원 준,[1] 김 현 철,[1] 엄 혜 민,[1] 이 지 나,[2] 권 혁 민,[3] 김 휘 강[3‡]
[1]엔씨소프트, [2]카카오, [3]고려대학교

A Study on Game Bot Detection Using Self-Similarity in MMORPGs

Eun-Jo Lee,[1†] Won-Jun Jo,[1] Hyunchul Kim,[1] Hyemin Um,[1] Jina Lee,[2]
Hyuk-min Kwon,[3] Huy-Kang Kim[3‡]
[1]NCsoft, [2]Kakao, [3]Korea University

요 약

다중 접속 역할 게임(Massively Multi-Online Role Playing Game, MMORPG)에서 게임봇은 게임 밸런스에 악영향을 끼치고 일반 유저들에게 상대적인 박탈감을 느끼게 하여 게임 수명을 단축시키는 위험 요소이다. 따라서 그 동안 게임 봇을 탐지하기 위한 다양한 방법이 연구되었으나 특정 게임 컨텐츠의 특징에 초점을 맞춤에 따라 신규 게임이 출시될 때마다 탐지 기법 개발이 필요하거나 혹은 게임 및 봇 프로그램 업데이트에 따른 유지 보수 방안을 고려하지 않고 있다. 본 논문에서는 게임봇이 본질적으로 갖고 있는 특징인, 설정된 패턴에 따라 행동을 반복하는 자기유사성을 주요 특질로 이용한 기계 학습 기법을 제안하고 이렇게 학습한 모델을 자동으로 유지 보수하는 시스템을 제안하였다. 이렇게 제안한 방법은 엔씨소프트의 대표 MMORPG인 리니지, 아이온, 블레이드 앤 소울에 대해 성능을 테스트하였으며 시스템을 구현하여 실전에 적용하였다.

ABSTRACT

Game bot playing is one of the main risks in Massively Multi-Online Role Playing Games(MMORPG) because it damages overall game playing environment, especially the balance of the in-game economy. There have been many studies to detect game bot. However, the previous detection models require continuous maintenance efforts to train and learn the game bots' patterns whenever the game contents change. In this work, we have proposed a machine learning technique using the self-similarity property that is an intrinsic attribute in game bots and automated maintenance system. We have tested our method and implemented a system to major three commercial games in South Korea. As a result, our proposed system can detect and classify game bots with high accuracy.

Keywords: game bot detection, self-similarity, online game security

I. Introduction

다중 접속 역할 게임(이하 MMORPG)에는 자동 사냥 프로그램을 사용하여 부정한 방법으로 게임을 플레이하는 유저(이하 게임 봇)가 존재한다. 자동 사냥 프로그램이란 게임 클라이언트 및 서버 로직을 역공학 기법을 이용해 분석하여 게임을 유저가 직접 컨트롤하지 않더라도 자동으로 플레이가 가능하게 만든 일종의 AI 프로그램이다. 게임 봇은 장기간 쉬지 않고 지속적인 게임 플레이가 가능하기 때문에 일반 유저보다 빠르게 성장하거나 더 많은 게임 내 재화를 획득할 수 있다. 따라서 적은 노력으로 쉽게 보상을 얻고자 하는 유저들이 이런 자동 사냥 프로그램을 사용한다. 특히, 상당수 온라인 게임은 게임 내에서 습득한 재화나 중요 아이템을 현금을 주고 사고파는 행위가 널리 퍼져 있으며 이런 게임 아이템 거래를 중

Received(06. 17. 2015), Modified(09. 14. 2015), Accepted(10. 05. 2015)
† 주저자, gimmesilver@ncsoft.com
‡ 교신저자, cendar@korea.ac.kr(Corresponding author)

개해 주는 사이트까지 존재하고 있다. 때문에 상업적인 목적을 갖고 대규모 게임 봇을 운영하는 일종의 사업체도 존재하는데 이를 보통 '작업장'이라고 부른다.

MMORPG는 여러 유저가 동일한 가상 세계 내에서 서로 경쟁하며 게임을 플레이하는데 이런 게임 봇은 정상적으로 게임을 즐기는 일반 유저보다 성장이나 재화 습득이 더 빨라 게임 내에서 제한된 여러 가지 자원을 독식할 수 있기 때문에 정상 유저들에게 상대적인 박탈감 및 불만을 느끼게 한다. 또한 게임 개발사 입장에서는 지나치게 빠른 컨텐츠 소비로 인해 기획 의도가 왜곡되고 더 나아가 게임 밸런스가 붕괴되어 장기적으로는 유저 이탈을 촉진하는 위험 요소가 된다. 특히 아이템 거래 사이트를 통한 현금 전환을 목적으로 대규모 사업화되어 게임 봇을 운영하는 작업장의 경우 게임 내 주요 지표에 큰 영향을 줄 수 있어 게임 운영상의 리스크(불확실성)로 작용한다. 따라서 사업/운영 부서에서는 유저 불만 해소 및 게임 컨텐츠의 빠른 소진 방지, 회사 이미지 제고 등의 목적으로 게임 봇 사용을 약관상에서 금지하고 있으며 이에 대한 관리 및 제재를 위해 노력하고 있다. 그리고 이를 위해선 게임 봇을 정확히 탐지하는 것이 선행되어야 한다.

게임 봇을 탐지하기 위해 보통 게임 회사에서는 게임 운영자나 데이터 분석가가 게임 봇을 사용하는 유저들의 플레이 특징을 찾거나 혹은 게임 봇 프로그램 역공학을 통해 프로세스 패턴을 찾아 이를 탐지 규칙으로 적용하는 방법을 많이 사용한다. 이 경우 탐지 규칙을 만들기까지 많은 시간과 노력이 필요하기 때문에 여러 개의 게임을 운영하는 회사 입장에서는 각 게임 별로 개별적인 탐지 규칙을 만들어야 하는 부담이 있다. 데이터 마이닝은 이렇게 탐지 규칙을 만드는데 드는 비용을 완화하는데 도움이 될 수 있다. 즉, 각 게임 유저가 플레이한 상세 이력이 담겨 있는 로그 데이터를 이용해 게임 봇과 일반유저를 구분할 수 있는 분류 모델을 학습하여 탐지에 사용하는 것이다. 그런데 분류 모델을 이용할 때는 어떤 특질을 사용하느냐가 매우 중요하며 이를 위해선 여전히 사람의 노력이 많이 필요하다. 또한 이렇게 탐지 규칙을 만든다 하더라도 게임 봇 프로그램 업데이트나 설정 변경을 통해 탐지 규칙을 우회할 수 있기 때문에 단기적인 효과는 거둘 수 있어도 장기적인 성과를 기대하기는 힘들다.

이를 해결하기 위해선 데이터 마이닝 기법을 이용할 때 가급적 특정 게임에 종속되지 않는 일반적인 특질을 찾아 여러 게임에 두루 적용할 수 있도록 게임 봇이 갖는 근원적인 속성을 찾아 활용해야하며, 이렇게 생성한 탐지 규칙을 지속적으로 유지하기 위한 유지 보수 기능을 자동화하는 것이 필요하다.

우리는 이를 위해 아래와 같은 두 가지 기법을 제안한다.

첫째, 게임 봇이 갖고 있는 근본적인 속성 중 자기 유사성에 주목하고 이를 특질로 이용하였다. 게임 봇은 정해진 설정에 따라 특정 플레이를 지속적으로 반복하기 때문에 일반 유저에 비해 동일한 행동을 반복할 가능성이 크다. 따라서 어떤 행위를 수행하는지에 상관없이 단지 자신이 과거에 수행한 행동을 얼마나 지속적으로 반복하는지를 일반적인 형태로 정량화할 수 있다면 이 특질은 여러 게임에서 두루 사용할 수 있다. 우리는 행위의 반복성을 일반적인 형태로 정량화하기 위해 자기 유사도를 측정하는 기법을 제안한다.

둘째, 이렇게 기계 학습을 이용해서 탐지 모델을 생성한다 하더라도 앞서 언급한 대로 게임 컨텐츠가 업데이트 되거나 혹은 탐지 규칙을 우회하기 위해 게임 봇이 변경될 경우 기존 탐지 모델이 더 이상 유효하지 않게 될 수 있다. 따라서 우리는 탐지 결과가 더 이상 유효하지 않게 되는 시점을 자동으로 탐지하고 모델 재학습을 통해 변경된 상황에 맞는 탐지 모델을 자동 생성하는 프로세스를 제안한다.

우리는 이 연구에서 제안한 탐지 기법이 여러 MMORPG에서 일반적으로 사용할 수 있는지 검증하기 위해 엔씨소프트의 대표 MMORPG인 리니지, 아이온, 블레이드 앤 소울에 대해 탐지 성능을 실험하였으며 라이브 시스템으로 구현하였다. 리니지의 경우 2015년 2월부터 적용하여 라이브 운영을 하고 있으며 아이온과 블레이드 앤 소울 역시 시스템 구현을 완료하여 현재 라이브 운영을 준비 중에 있다.

II. Related Works

기계 학습을 이용하여 게임 봇을 탐지하기 위한 다양한 연구가 있다. M.A.Ahmad 등[1]은 '에버퀘스트 2'라는 게임을 대상으로 다양한 분류 알고리즘과 특질 셋을 사용하여 성능을 측정한 선구적인 연구이다. 게임 봇을 gatherer, banker, dealer, marketer 로 유형화했으며 탐지 모델에 사용할 특질로 demographic data, sequence data, statistical activity, economic transaction,

network centrality 등 다양한 데이터를 사용하였다. 그러나 지나치게 일반적인 특질을 사용하였고 학습 및 테스트를 위한 신뢰도 높은 데이터를 사용하지 못해 탐지 성능이 매우 낮은 문제가 있었다. Thawonmas 등[2]은 게임 봇이 일반 유저에 비해 특정 행동을 반복 수행한다는 특성을 이용하여 통계적인 관점에서 분류를 시도한 거의 최초의 연구이다. 그러나 단순히 액션 타입 별 빈도수 통계에 의존했기 때문에 단순한 행동만 반복하는 게임 봇을 탐지하는 데에는 효과적일지 모르나 최근에 사용되고 있는 높은 수준의 게임 봇을 탐지하기에는 한계가 있다.

반면 일반적인 특성을 이용한 탐지 기법이 대개 성능이 좋지 못하다는 점을 보완하기 위해 특정 행위에 집중해서 게임 봇을 탐지하는 연구들도 있다. Kang 등[3]은 MMORPG에서 캐릭터 간에 주고받는 채팅 데이터에 주목하여 게임 봇과 일반 유저를 분류하는 기법을 제시하였으며 또 다른 연구[4]에서는 파티 플레이 특징을 이용한 봇 탐지 기법을 제안하였다. 이렇게 게임 봇의 특정 행위에 기반을 둔 탐지 기법의 경우 세밀한 탐지가 가능하기 때문에 높은 성능을 기대할 수 있는 장점이 있으나 각 게임별 제공 컨텐츠 및 게임 봇 프로그램의 특성에 의존하기 때문에 다양한 게임에 적용하기 어렵고 별도의 튜닝 과정이 필요한 단점이 있다.

한편, 게임 봇이 다양한 행동 유형을 가질 수 있기 때문에 이런 행동 유형별로 각기 다른 탐지 모델을 적용하기 위한 시도도 있다. Y. Chung 등[5]은 다양한 게임 로그를 이용해 각 캐릭터들을 행동이 유사한 그룹별로 군집화하고 각 그룹에 대해 별도의 탐지 모델을 적용하는 기법을 제안하였다. 이 기법은 기존 탐지 기법에 비해 좀 더 정교하고 일반적인 탐지를 수행할 수 있으나 각 유형별로 학습 집합을 생성해야하기 때문에 탐지 모델 생성에 높은 비용이 든다. 또한 게임 컨텐츠 업데이트로 인해 플레이 패턴이 바뀌면 행동 유형을 다시 군집화하고 학습 집합을 생성해야 하는 부담이 있다.

반면 게임 봇의 일반적인 특성을 찾기 위한 연구도 있다. Chen 등[6] 과 Kesteren 등[7]은 캐릭터의 이동 경로를 이용한 탐지 기법을 제안하였다. 이 두 기법은 게임 봇과 같은 자동 프로그램의 경우 일반 유저에 비해 미리 설정된 정보에 의해 일정한 경로로 움직이는 경향이 더 높다는 특성을 이용한 탐지 기법으로써 대부분의 MMORPG에서 적용 가능한 기법이라는 장점이 있다. 그러나 정밀한 이동 경로 파악을 위해선 로그의 정밀도를 높여야 하는데 이는 시스템에 큰 부하를 줄 수 있어 실전에 적용하기 힘들다. 또한 게임봇이 이동 경로에 임의성을 갖도록 프로그래밍이 될 경우 충분히 우회가 가능하다는 단점도 갖고 있다.

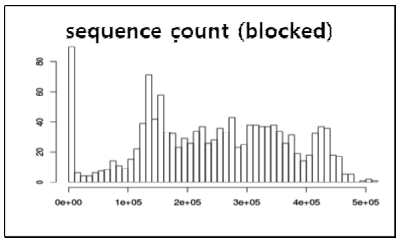
우리가 사용한 자기 유사도 알고리즘은 이동 경로 기반 탐지 기법처럼 게임 봇의 범용적인 특성인 반복성을 이용한 방식이다. 다만 이동 경로 기반 탐지 기법의 경우 반복성 측정 대상이 이동 경로였는데 이는 지나치게 정밀한 측정 대상이기 때문에 앞서 언급했듯이 약간의 임의성을 줌으로써 우회가 상대적으로 용이한 반면 우리가 사용한 자기 유사도 알고리즘은 MMORPG 내에서 수행 가능한 캐릭터의 모든 행동을 대상으로 삼았기 때문에 상대적으로 우회 패턴을 만들기 어려워 견고하다는 장점이 있다.

이러한 자기 유사성은 통신 트래픽 분석 분야에서 많이 연구되었는데 특히 Crovella 등[8]의 연구가 가장 널리 알려져 있다. Kwon 등[9]은 트래픽의 자기 유사성을 응용하여 침입을 탐지하는 방법을 제안하였고 Cevizci 등[18]은 온라인 게임 트래픽 분석에 적용하여 서버와 클라이언트에서 각각 트래픽의 자기 유사성을 비교 분석하였다.

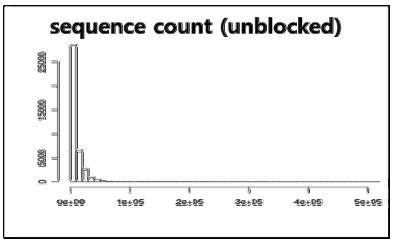
한편, 앞선 자기 유사성 관련 연구는 네트워크 트래픽을 대상으로 한 것에 비해 Kwon 등[10]은 게임 로그를 대상으로 분석을 진행하여 게임 봇과 일반 유저를 분류하는데 자기 유사도를 사용할 수 있는 가능성을 제시하였다. 우리는 [10]의 연구에서 제시한 자기 유사도 측정 방식이 갖는 한계점(이 한계점에 대해서는 4.1.4에서 언급)을 보완하기 위해 보조 특질을 더 추가했으며, 기존 연구에서는 게임 봇과 일반 유저를 구분하기 위한 명확한 기준을 제시하지 못한 점을 극복하기 위해 탐지 모델 생성 시 신뢰도 높은 정답 집합을 구축하여 학습 및 평가에 사용하였다. 또한 [10]에서는 아이온에 대해서만 한정하여 실험하였으나 이번 연구에서는 아이온뿐만 아니라 리니지와 블레이드 앤 소울에 대해서도 실험을 실시하였으며 이를 통해 본 연구에서 제안한 알고리즘이 여러 MMORPG 에서 두루 쓰일 수 있다는 것을 보였다.

III. Self-similarity of Game Bots

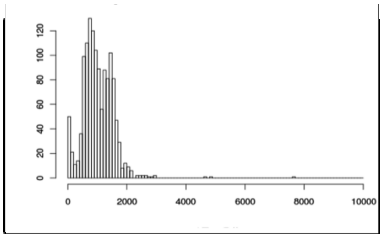
우리는 게임 봇이 일반 유저에 비해 특정 행동을 반복 수행하는 경향이 더 강한지를 확인하기 위해 데이터 분석을 수행하였다. 이를 위해 리니지, 아이온,



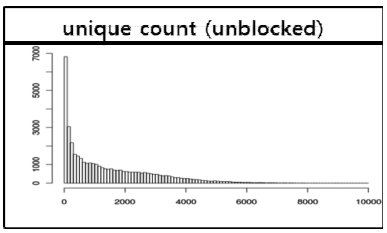
(a) total count per blocked user



(b) total count per unblocked user



(c) unique count per blocked user



(d) unique count per unblocked user

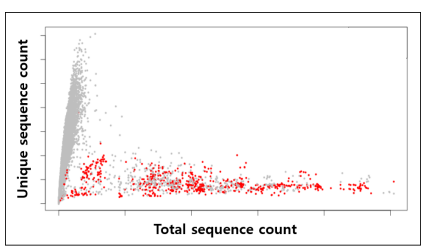
Fig. 1. Histogram of sequence data

블레이드 앤 소울에서 과거에 게임 봇을 사용하여 제재를 당한 계정 목록을 입수했다. 그리고 이 계정들과 일반 계정의 게임 로그 데이터를 이용해서 이 두 집단 간의 차이를 비교 분석하였다.

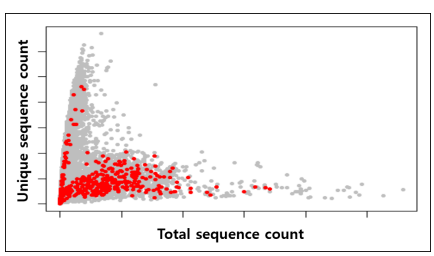
게임 로그는 유저들이 사냥, 채집, 아이템 거래, 게임 접속 및 종료 등의 행동을 수행하거나 특정 이벤트가 발생할 때마다 게임 서버에 기록된다. 이 각각의 이벤트나 행동은 그 종류에 따라 고유 아이디가 있다. 이를 로그 아이디라 부른다. 우리는 유저별로 이 로그 아이디의 리스트를 시퀀스 데이터로 생성했다. 그리고 이 전체 시퀀스를 일정 개수 단위로 이동해가며 부분 집합 시퀀스를 만들었다. Fig.1은 이렇게 생성된 계정별 시퀀스 데이터의 종류와 개수에 대한 분포를 제재 계정과 일반 계정으로 분류하여 비교한 그래프이다.

Fig.1 에 나와 있듯이 제재 유저는 일반 유저에 비해 시퀀스 개수가 높은 쪽에 많이 분포되어 있는 반면 시퀀스의 종류 수는 상대적으로 높지 않다. Fig.2 는 이 특징을 좀 더 명확히 보여 준다. Fig.2 를 보면 전체 유저는 시퀀스 총 개수 대비 시퀀스 종류 수의 비율에 따라 크게 2개 집단을 형성하고 있는데 대부분의 제재 계정은 시퀀스 개수 대비 종류 수의 비율이 낮은 쪽 집단에 포함되어 있다. 이것은 제재 유저들이 일반 유저에 비해 특정 행동을 반복적으로 수행하는 것을 의미한다.

이처럼 시퀀스 데이터가 게임 봇을 일반 유저와 구분하는 좋은 데이터이긴 하지만 전체 게임 유저에 대해서 시퀀스 데이터를 생성하는 것은 시스템 처리 비용이 큰 작업이다. 예를 들어 각 유저별로 N개의 게임 로그 데이터가 있을 경우 이것을 k개씩 묶음으로 시퀀스 데이터를 생성하기 위해 $N\times(N-k)$ 만큼의 데이터를 생성해야 한다. 또한 데이터를 시간 순으로 정렬해야 하는데 한 번에 처리해야 하는 게임

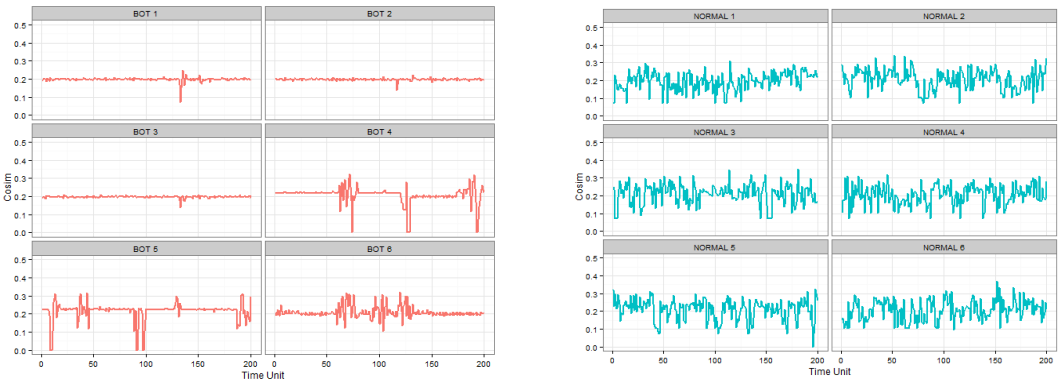


(a) B&S



(b) Aion

Fig. 2. Scatter plot comparing unique count with total count of log sequence per user: blocked users are red, non-blocked users are gray.



(a) bots (b) normal users

Fig. 3. Time series plot of cosine similarity

로그의 수는 약 수억 ~ 수십억 개 수준이기 때문에 분산 시스템을 통해 처리한다고 하더라도 정렬에 드는 계산 비용이 매우 크다.

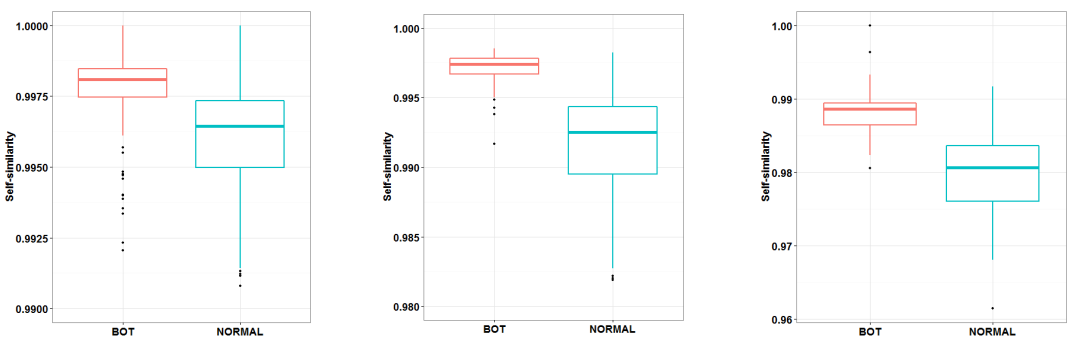
따라서 우리는 이보다 더 적은 처리 비용으로 게임 봇의 자기 유사성을 표현할 수 있는지 확인해 보기 위해 시퀀스 대신 벡터를 이용하여 분석을 진행했다.

먼저 게임 로그를 특정 시간 단위로 묶어서 벡터를 생성한 후 각각의 벡터가 서로 얼마나 유사한지 측정하기 위해 각각의 벡터와 단위 벡터(벡터의 크기가 1인 벡터)간의 코사인 유사도를 측정했다. 만약 특정 유저가 비슷한 행동(시퀀스)를 반복한다면 유사한 벡터가 반복해서 생성될 것이고 결국 단위 벡터와 계산된 각각의 코사인 유사도 역시 비슷한 값이 반복적으로 생성될 것이다. 그러면 우리는 이 코사인 유사도만을 갖고도 유저가 특정 행동을 반복한다는 것을 개략적으로 확인할 수 있다.

Fig.3 은 같은 기간에 대해서 게임 봇과 일반 유저의 코사인 유사도 변화를 비교한 시계열 차트이다. 그림에서 보다시피 일반 유저는 시간의 흐름에 따라 코사인 유사도 값의 변화가 큰 반면 게임 봇은 일부 구간을 제외하고는 대체로 비슷한 값을 유지한다.

마지막으로 우리는 이 코사인 유사도 값들을 이용해 자기 유사도를 측정했다. Fig.4 는 게임 봇과 일반 유저의 자기 유사도 값을 비교한 박스 플롯이다. 그림에 나와 있듯이 우리가 실험한 세 개 게임 모두 게임 봇이 일반 유저에 비해 자기 유사도가 높은 경향을 보인다.

따라서 시퀀스 데이터를 생성하지 않고 대신 벡터를 이용하더라도 자기 유사도를 측정할 수 있으며 이를 통해 게임 봇과 일반 유저를 충분히 구분할 수 있다. 특히 우리가 제안한 방식(세부적인 알고리즘은 4.1에서 설명)에서는 벡터 생성 시 로그의 정렬이 필요 없고 단지 유저별 시간별 분류 후 로그 아이디별 발생 개수를 세는 작업만 필요하기 때문에 시퀀스



(a) Lineage (b) Aion (c) B&S

Fig. 4. Comparison of self-similarity between bots and normal users

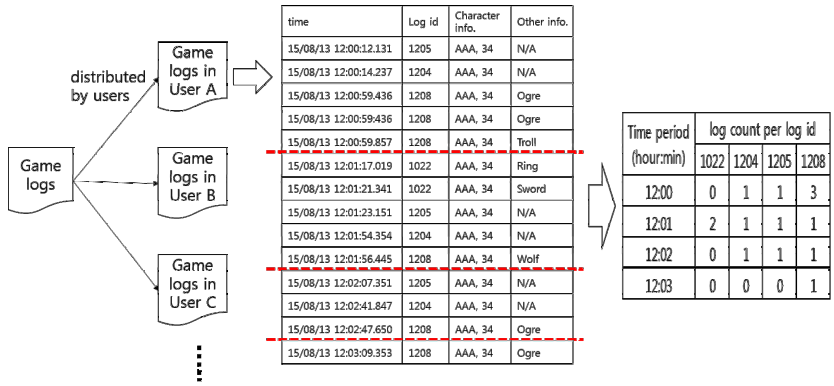


Fig. 5. Process for transforming game logs to vectors. In the above, four vectors are generated: (0,1,1,3), (2,1,1,1), (0,1,1,1), (0,0,0,1)

데이터에 비해 계산 비용 및 결과 데이터 크기가 훨씬 작다.

IV. Methodology

우리는 게임 봇을 탐지하기 위한 전체 프로세스 및 알고리즘을 제안한다. 4.1에서는 특질 데이터 추출 방법에 대해서 소개하며 4.2 에서는 탐지 모델을 생성하고 이를 평가하기 위해 우리가 사용한 방법에 대해 설명한다. 마지막으로 4.3에서는 생성된 모델의 성능을 모니터링하고 변화 감지 시 자동으로 모델을 수정하는 프로세스를 설명한다.

4.1 Feature selection

앞서 언급했듯이 우리는 탐지 모델 생성을 위해 유저별 자기 유사도를 측정하였다. 자기 유사도 측정 알고리즘은 다음과 같다.

4.1.1 로그 벡터 생성

서버에서 수집된 게임 로그를 각 캐릭터별로 분류하고 시간 순으로 정렬한다. 이후 분류된 로그를 다시 일정 시간 단위로 묶는다. 이렇게 묶인 로그에 대해서 로그 벡터를 생성하는데, 벡터의 차원은 해당 게임에서 나올 수 있는 전체 로그 아이디가 되며 각 차원의 값은 해당 차원의 로그 아이디가 해당 시간대에 발생한 횟수가 된다. Fig.5 는 여기서 설명한 로그 벡터를 생성하는 절차를 설명한 그림이다.

4.1.2 로그 벡터와 단위 벡터 간 코사인 유사도 측정

앞 단계에서 생성한 로그 벡터 각각에 대해 단위 벡터(크기가 1인 벡터)와의 코사인 유사도를 측정한다. 코사인 유사도 공식은 Eq.1 과 같다.

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A)^2} \times \sqrt{\sum_{i=1}^{n} (B)^2}} \quad (1)$$

4.1.3 코사인 유사도의 표준 편차 측정

2단계에서 구한 유저 별 코사인 유사도의 표준 편차(δ)를 Eq. 2 에 적용해 자기 유사도 지수(H)로 변환한다. H는 자기 유사성이 높을수록 1에 근접하며 반대의 경우 0.5에 가까워진다. 이렇게 변환한 값은 Hurst 지수*와 동일한 스케일을 갖게 된다.

$$H = 1 - \frac{1}{2} \times \delta \quad (2)$$

4.1.4 추가 특질 적용

만약 캐릭터가 아무런 행동을 하지 않고 장시간 대기하고 있거나 혹은 극히 짧은 시간만 플레이하는 경우 자기 유사도가 높게 나오는 문제가 있다. 따라서 우리는 이런 점을 보완하기 위해 보조 특질을 추가했다. Table 1. 은 우리가 사용한 전체 특질을 정

* https://en.wikipedia.org/wiki/Hurst_exponent

Table 1. List of features used in the proposed detection method

no	name	description
1	self_sim	self similarity
2	vector_count	count of a set of log vectors
3	uniq_vector_count	unique count of a set of log vectors
4	cosim_zero_count	count of data in which cosine similarity is zero in a set of log vectors
5	vector_mode	count of data that appears most often in a set of log vectors
6	total_log_count	total count of logs generated by user
7	char_level	character level
8	play_time	play time during certain period per user
9	npc_kill_count	NPC kill count
10	trade_take_count	count of trade in which user takes items from another
11	trade_give_count	count of trade in which user gives items to another
12	retrieve_count	count of activity in which user retrieves items from warehouse
13	deposit_count	count of activity in which user deposits items to warehouse
14	log_count_per_min	average count of logs are generated per minute

리한 표이다.

4.2 탐지 모델 생성 및 평가

우리는 로지스틱 회귀 모델을 이용하여 게임 봇 탐지 모델을 생성한다. 로지스틱 회귀 모델을 이용하면 각 특질별 회귀 계수가 나오는데 이 각각의 회귀 계수를 Eq. 3 에 적용하여 캐릭터별 게임 봇 확률을 계산 한다(α와 β는 회귀모델에서 나오는 회귀 계수를 의미한다).

$$P_{BOT} = \frac{1}{1+e^{-(\alpha+\sum_{i=1}^{n}\beta_i x_i)}} \quad (3)$$

또한 K-fold cross validation을 이용해 탐지 모델의 성능을 평가한다. K-fold cross validation은 정답 집합을 K개의 부분 집합으로 나눈 후 첫 단계에서는 첫 번째 집합을 테스트 집합, 나머지를 학습 집합으로 이용하여 학습 및 검증을 수행하고 그 다음 단계에서는 두 번째 집합을 테스트 집합, 나머지를 학습 집합으로 이용하여 학습 및 검증을 수행하는 식으로 총 K번의 검증을 통해 모델의 성능을 측정한다. 그리고 이렇게 측정된 K개의 측정치에 대한 평균이 모델의 최종 성능이 된다.

검증 시에는 Receiver Operating Characteristic (ROC) 곡선에 대한 면적(Area Under the Curve, AUC)을 구하여 성능을 측정한다. ROC 곡선은 이진 분류기 성능 측정 방법 중 하나로, threshold 변화에 따른 오탐율(False Positive Rate, FPR)과 재현율(True Positive Rate, TPR)을 각각 x, y좌표로 표시한 그래프이다. TPR과 FPR은 다음과 같이 구한다.

$$TPR = \frac{TP}{TP+FN},\ FPT = \frac{FP}{FP+TN} \quad (4)$$

Eq. 4.에서 TN과 FN은 각각 False Negative와 True Negative를 의미한다.

일반적으로 threshold값이 커질수록(즉, 기준이 엄격해질수록) 재현율과 오탐율은 낮아지고 반대일수록 높아진다. 따라서 threshold를 낮춰서 재현율이 높아지더라도 상대적으로 오탐율이 낮게 유지되면 좋은 모델이라 할 수 있으며 이 경우 ROC 곡선은 왼쪽 상단으로 불룩한 그래프가 되어 AUC 가 커진다.

4.3 결과 모니터링 및 탐지 모델 재학습

우리는 게임 컨텐츠 업데이트나 게임 봇 행동 변화로 인해 기존에 학습한 탐지 모델이 유효하지 않게 되는 경우 이 시점을 자동으로 감지하기 위한 프로세스를 제안한다. 이를 위해 전체 계정의 게임 봇 확률을 정기적으로 측정하여 이 값이 과거에 비해 큰 폭으로 변하는 경우 탐지 모델을 자동으로 갱신하고 갱

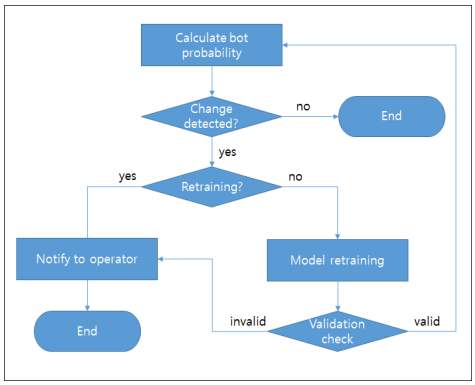


Fig. 6. Automated maintenance process

신한 탐지 모델이 적합한지 검증한다. 만약 재학습한 탐지 모델 역시 적절하지 않은 경우 시스템은 자동으로 운영자에게 해당 사실을 통보한다. 이 전체 프로세스는 Fig. 6 과 같다.

우리는 Fig.6 의 프로세스 중 '변화 감지(Change detected)' 단계에서 EWMA 라는 기법을 사용했는데 세부 알고리즘은 다음과 같다[11].

4.3.1 게임봇 확률 상관 계수 계산

유저 i 에 대해 t 시점에서의 게임 봇 확률을 a_i, $t-1$ 시점에서의 게임 봇 확률을 b_i 라고 할 때 t 시점에서의 게임 봇 확률 상관 계수 x_t는 아래 식처럼 계산한다. 이 때, $t-1$ 시점에서 게임 봇 확률이 없는 신규 유저나 반대로 t 시점에 활동 내역이 없는 이탈 유저는 계산에서 제외한다.

$$x_t = \frac{n\sum a_i b_i - \sum a_i \sum b_i}{\sqrt{n\sum a_i^2 - (\sum a_i)^2}\ \sqrt{n\sum b_i^2 - (\sum b_i)^2}} \qquad (5)$$

4.3.2 상관 계수 변화량 계산

위 단계에서 측정한 상관계수에 대해 가중치 이동평균을 계산한다. 이 때 가중치 λ를 정해야 한다. λ 값이 클수록 가장 최근에 측정한 상관 계수 x_t의 비중을 크게 두고 평균을 계산하게 된다. 따라서 λ를 적절하게 정하는 것이 중요한데 이를 정하는 방식에 대해서는 5.4에서 설명한다.

$$z_t = \lambda x_t + (1-\lambda) z_{t-1},\ 0 < \lambda \leq 1,\ t = 1,2,\ldots \qquad (6)$$

4.3.3 관리 한계치 측정 및 비교

위 단계에서 계산한 z_t 값이 관리 한계치를 벗어나는 경우 게임 봇 패턴에 변화가 생겨 계정들의 확률값을 재계산해야 한다고 판단한다. 관리 한계치는 상한선과 하한선이 있으며 아래 식을 이용해 기준을 정한다. 이 식에서는 L 상수를 정해야 하는데, L 이 작을수록 작은 변화도 민감하게 탐지하게 되며 L 이 크면 대부분의 상황을 정상으로 간주하게 된다. L 값을 정하는 방법에 대해서는 5.4에서 설명한다.

$$CL = \mu \pm L\delta\sqrt{\frac{\lambda}{z-\lambda}},\ \mu = z\text{의평균},\ \delta = z\text{의표준편차} \qquad (7)$$

4.3.4 모델 검증

관리 한계치를 넘어선 경우 모델 재학습을 수행한다. 재학습한 모델은 시스템에 적용하기 전에 적절성을 검증한다. 검증은 K-fold cross validation을 이용하는데 AUC가 0.9 이하인 경우 부적합하다고 판단한다. 재학습 결과가 부적절하다고 판정된 경우 시스템은 운영자에게 해당 사실을 통보하고 탐지 프로세스를 종료함으로써 신뢰도가 떨어지는 탐지 결과를 사용하여 생길 수 있는 문제를 미연에 방지한다.

V. Experiments

5.1 정답 집합 구축

게임 봇 탐지 모델을 구축하기 위해선 학습 및 검증에 필요한 정답 집합(ground truth)이 필요하다. 이전 연구들에서는 기존에 다른 방법을 이용해서 탐지한 데이터를 정답 집합으로 사용하거나[1], 실험을 위해 사람과 봇 프로그램을 이용해 인위적으로 생성한 데이터를 이용하였다[2]. 그러나 이 경우 미탐 데이터에 대한 정보를 알 수 없기 때문에 정확한 학습 및 성능 측정이 불가능한 문제가 있다. 그래서 Ahmad 등[12]은 생태학에서 개체의 규모를 추정하는데 사용하는 capture-recapture 방법과 그래프 분석을 이용하여 이런 문제를 완화하는 기법을 제안하기도 했다. 그러나 여전히 정확한 정답 집합 부재로 인한 문제는 남아 있다. 그래서 우리는 이 제약을 극복하기 위해 비용이 다소 많이 들더라도 사람이 육안 검사를 통해 게임 봇과 일반 유저를 구분하는

방법을 이용해 정답 집합을 구축하였다.

MMORPG에서는 게임 내 고객 지원 및 모니터링을 위해 사용하는 운영 목적의 게임 캐릭터가 있다. 이를 'GM캐릭터'라고 하는데 이 캐릭터는 일반 유저에게 보이지 않으며 일반 유저는 할 수 없는 특수 기능을 수행할 수 있다. 이런 특수 기능 중에 '순간 이동' 기능이 있는데 이를 이용하면 지정한 캐릭터가 현재 존재하는 지역으로 이동하여 그 캐릭터의 행동을 관찰할 수 있다. 우리는 정답 집합 구축 시 이 기능을 이용해 미리 정한 표본 집합에 대한 관찰을 통해 게임 봇 여부를 판단하였다. 이렇게 육안 검사를 실시할 경우 유저의 세밀한 동작을 관찰할 수 있기 때문에 숙련된 운영자는 육안 검사를 통해 캐릭터가 게임 봇인지 여부를 매우 높은 정확도로 판단할 수 있다. 다만 실수나 주관적인 편향에 의해 잘못된 정답 집합을 구축할 수 있기 때문에 이런 문제를 방지하기 위해 아래와 같은 규칙 및 프로세스를 만들었다.

첫째, 정답 집합 구축 시 최소 2명 이상의 검사자가 동일한 시간에 동일한 캐릭터에 대해 각각 독립적으로 판정 작업을 진행한 후 결과를 취합한다. 그리고 취합한 결과 중 검사자 모두 동일한 판정을 한 표본만 정답 집합으로 사용하였다.

둘째, 게임 봇 판정 시 판정 사유를 기입하게 하였다. 이렇게 기입한 판정 사유는 사후 검증 작업을 통해 정형화하여 다음 판정 작업 수행 시 참고하였다.

셋째, 정답 집합을 생성한 후에 각각의 판정 대상에 대해서 Leave One Out Cross Validation (LOOCV)을 수행하였다. LOOCV는 정답 집합에 대해 1개를 제외한 나머지 데이터를 학습 집합으로 이용하여 모델을 생성한 후 이 모델을 남은 1개 데이터에 적용하여 예측값과 실제값이 차이가 나는지 검증하는 기법이다. 이 검증 프로세스는 구축한 정답 집합 적용 직전에 수행되며 만약 예측값과 검사자가 입력한 값에 차이가 날 경우 검사자에게 전달되어 재검사를 수행하게 된다.

위에서 설명한 세 가지 규칙 중 앞에 두 가지는 특정 검사자의 주관적인 편향에 의해 게임 봇을 잘못 판정할 가능성을 최소화하기 위함이며 세 번째 규칙은 검사자가 최종 결과 입력 시 실수로 게임 봇을 일반 유저로 기입하거나 혹은 그 반대의 상황이 발생하는 일을 방지하기 위함이다.

우리는 이 정답 집합 구축 프로세스를 이용해서 리니지(Lineage), 아이온(Aion), 블레이드 앤 소울(B&S)의 정답 집합을 구축하였다. 2013년 12월부터 2015년 3월까지 운영자 2~3명이 작업했으며 각 게임별로 여러 차례에 걸쳐 점진적으로 구축을 진행하였다. 구축한 정답 집합 목록은 Table 2. 에 정리했다.

Table 2. Ground Truth Construction Results

Game	Bot	Human
Lineage	315	244
Aion	75	75
B&S	260	230
Total	650	549

5.2 데이터 전처리

우리는 정답 집합을 구축한 세 개 게임에 대해서 특질 데이터를 추출하기 위한 데이터 전처리 로직을 개발하였다. 비록 학습 대상이 되는 캐릭터는 1199개 이지만 라이브 시스템 개발을 고려했기 때문에 전체 사용자를 대상으로 추출 작업을 수행했다. 발생 가능한 로그 종류는 각 게임별로 200~300 종류가 되지만 정확한 자기 유사도 측정을 위해선 캐릭터 플레이와 직접적으로 연관된 게임 로그만 추출해서 벡터를 생성하는 것이 적절하기 때문에 게임 별로 처리할 게임 로그 종류를 제한하였다. 각 게임별 사용한 로그 종류 및 데이터 규모는 Table. 3 에 정리하였

Table 3. Summary of raw data (B: Billions)

Game	Logs	Types of logs	Users
Lineage	28.5 B	165	500K
Aion	5.0 B	229	250K
B&S	6.9 B	109	130K

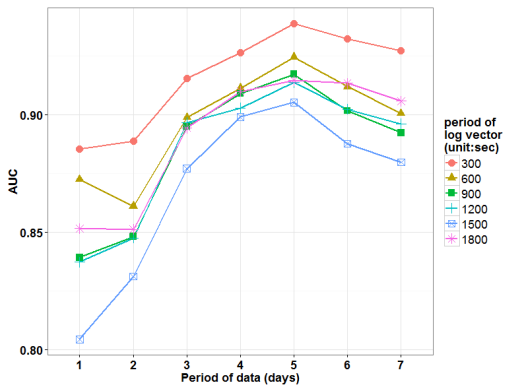


Fig. 7. AUC of detection model per vector period

다(전체 로그 개수와 유저 수에 대한 집계 기준은 일주일이며 한 달 데이터에 대한 평균값을 취했다).

데이터 전처리 시 두 가지 주요 파라미터를 결정해야 한다. 첫째, 특질 데이터를 생성할 때 집계할 데이터 기간과 둘째, 자기 유사도 측정을 위해 사용하는 로그 벡터의 시간 단위이다. 우선 데이터 집계 기간의 경우 대상 게임들은 게임 서버 점검 및 업데이트를 위해 매주 수요일 새벽에 다운타임을 갖기 때문에 대부분의 플레이는 일주일 주기로 패턴이 변경된다. 따라서 우리는 특질 데이터 생성 시 일주일 데이터를 취합하는 것이 가장 적절하다고 생각했다. 자기 유사도 측정을 위한 로그 벡터를 생성하는 시간 단위는 너무 짧을 경우 하나의 벡터에 담기는 정보량이 너무 적게 되고 반대로 너무 길면 벡터의 개수 자체가 너무 적어서 정밀한 유사도 측정이 어려워진다. 따라서 우리는 최적의 벡터 주기를 결정하기 위해 300초 ~ 1800초 범위에서 300초 단위로 자기유사도만을 이용하여 탐지 모델을 생성한 후 가장 AUC가 높은 주기를 사용하였다(Fig.7). Fig.7 에서 X축은 자기 유사도 측정 시 사용한 데이터 기간, Y축은 K-fold cross validation을 이용한 AUC 이며 로그 벡터 주기 별로 꺾은선 그래프로 값을 표시했다. 이 실험에 의하면 로그 벡터의 주기는 짧을수록 좋고, 데이터 기간은 5일이 가장 좋다. 그러나 앞서 언급했듯이 1주일 단위 정기 점검에 맞추는 것이 시스템 관리에 더 유리하고 심지어 5일과 7일의 성능 차이가 크지 않기 때문에 데이터 기간은 7일로 정하였으며, 로그 벡터 주기는 300초로 정하였다. 참고로 실험 데이터를 보면 로그 벡터 주기를 더 짧게 할 경우 AUC가 더 높아질 가능성이 있지만 로그 벡터 주기를 짧게 할수록 전처리에 필요한 시스템 리소스가 많이 필요하기 때문에 더 이상의 튜닝은 진행하지 않았다.

5.3 Modeling & Evaluation

우리는 게임 봇 탐지를 위한 기계 학습 알고리즘으로 로지스틱 회귀 모델을 이용하였다. 각 게임별로 학습을 통해 생성한 회귀 모델 결과는 Table. 4와 같다. 각 게임별로 통계적 유의성을 갖는 특질들이 조금씩 다르지만 자기 유사도(self sim) 변수는 공통적으로 통계적 유의성을 갖고 있다(p-value가 0.05보다 작다). p-value가 비록 변수의 유용성을 확증하는 절대적인 지표는 아니지만 적어도 우리가

Table. 4. Regression Coefficient of Features

variable	reg. coeff.	z-value	p-value
self sim.	**-3.757e+00**	**-2.686**	**0.00724 ****
total log count	-9.493e-06	-1.488	0.13672
main char. level	**-8.391e-02**	**-3.108**	**0.00188 ****
total use time min.	4.377e-03	1.063	0.28798
npc kill count	-1.176e-03	-0.169	0.86585
trade get count	-4.877e-02	-1.123	0.26133
trade give count	1.056e-02	0.247	0.80461
retrieve count	**1.582e-02**	**3.003**	**0.00267 ****
deposit count	**-1.040e-02**	**-2.175**	**0.02961 ***
log count per min.	1.808e-02	1.168	0.24267

(a) Lineage

variable	reg. coeff.	z-value	p-value
self sim.	**1.337e+03**	**2.651**	**0.00802 ****
total log count	-8.542e-05	-0.667	0.50470
main char. level	**-1.809e-01**	**-2.257**	**0.02399 ***
total use time min.	-1.550e-04	-0.024	0.98062
npc kill count	**-2.551e-04**	**-2.546**	**0.01091 ***
trade get count	3.648e-03	0.240	0.81056
trade give count	-8.576e-03	-0.233	0.81541
log count per min.	-1.371e+00	-1.334	0.18235

(b) Aion

variable	reg. coeff.	z-value	p-value
self sim.	**-6.469e+00**	**-2.251**	**0.02439***
total log count	**1.026e-04**	**3.143**	**0.00167 ****
main char. level	1.002e+00	0.038	0.96995
total use time min.	3.965e-05	0.053	0.95735
npc kill count	1.540e-05	0.100	0.92023
trade get count	-7.175e-04	-0.471	0.63796
trade give count	-3.292e-04	-0.034	0.97300
log count per min.	-2.050e-02	-0.768	0.44278

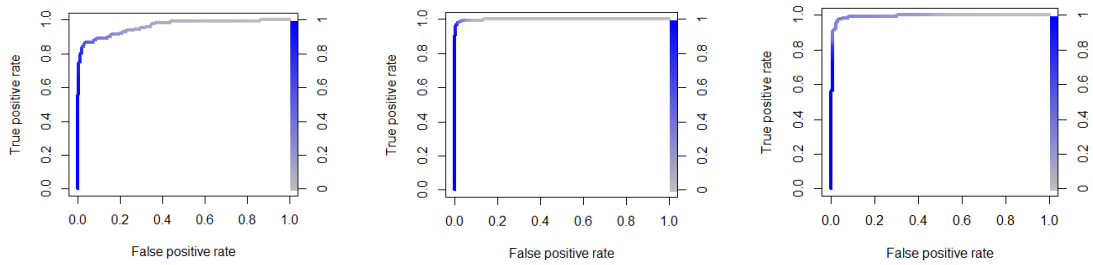
(c) B&S

실험한 세 개의 MMORPG에서 공통적으로 통계적 유의성을 보여준 점을 통해 우리는 이 연구에서 제안한 자기 유사도가 여러 MMORPG에서 일반적으로 사용할 수 있는 특질이 될 수 있음을 간접적으로 보여주는 자료라고 생각한다.

탐지 모델의 성능은 K-fold cross validation을 이용해 검증하였는데 이때 K는 10으로 정하였다. 게임 별 탐지 모델 평가 결과는 Table 5와 Fig. 8에 나와 있다. Table 5. 에서 검증에 사용한 게임

Table. 5. Performance of detection model

Game	Bot	Human	AUC
Lineage	148	145	0.9197
Aion	64	70	0.9931
B&S	163	143	0.9915



(a) Lineage (b) Aion (c) B&S

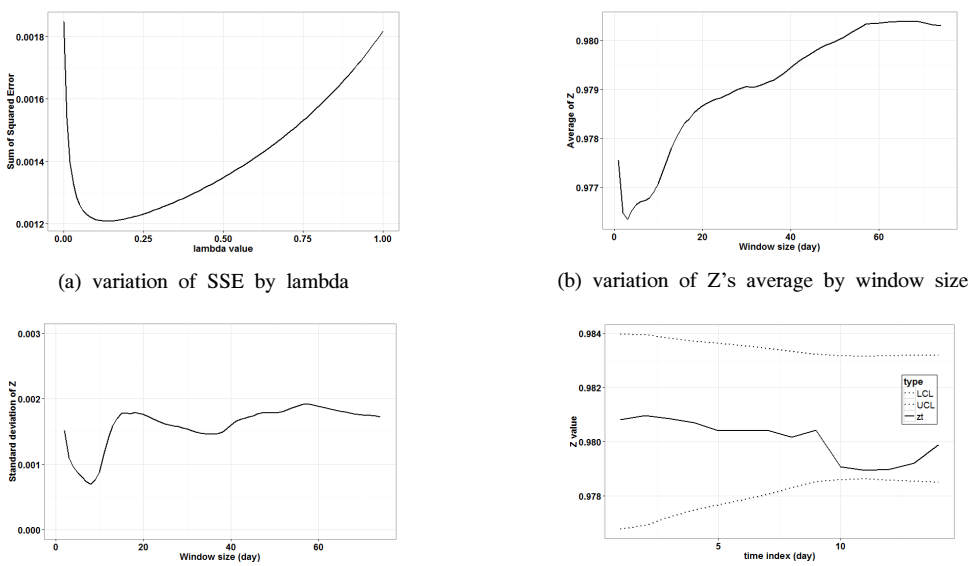
Fig. 8. ROC curve of detection model

봇과 일반유저의 수가 구축한 정답 집합 개수 (Table 2)와 차이를 보이는 이유는 정답 집합 상의 캐릭터들이 검증 시점에 플레이를 하지 않는 경우 때문이다.

5.4 Automated Model Maintenance

우리는 개발한 탐지 모델이 충분한 신뢰성을 갖고 있다고 판단했기 때문에 이 탐지 모델을 이용해 지속적으로 유저들의 게임봇 확률을 계산하고 규모를 파악하는 라이브 시스템을 구현하였다. 그런데 장기간에 걸쳐 지속적으로 게임봇을 탐지할 경우 게임 컨텐츠 업데이트로 인해 유저의 플레이 패턴이 바뀔 수 있고 게임 봇 사용자 역시 탐지를 우회하기 위해 행동 패턴을 변경하거나 봇 프로그램을 업데이트하기 때문에 이로 인해 탐지 모델이 노후화될 수 있다. 비록 우리가 제안한 탐지 모델은 행동 패턴에 최대한 덜 영향을 받는 특질을 사용하지만, 탐지 모델에 사용되는 파라미터 값 자체는 시간이 지남에 따라 최신 데이터에 적합하지 않을 수 있다. 따라서 모델이 노후화되면 최신 특질 데이터를 이용해서 모델 재학습 과정을 통해 파라미터를 수정해주는 것이 바람직하다. 이를 위해 우리는 모델이 노후화된 시점을 감지하고 이 시점에 탐지 모델을 자동으로 재학습하는 모듈을 시스템에 적용하였다.

전체 과정은 Fig.6 에 나와 있으며 변경 시점을 감지하기 위한 알고리즘은 4.3에서 설명하고 있다. 그런데 이 알고리즘을 구현하기 위해서는 가중치 평



(a) variation of SSE by lambda (b) variation of Z's average by window size

(c) variation of Z's std. deviation of window size (d) variation of Z and control limit by date

Fig. 9. EWMA parameter tuning

균(z)을 구할 때 필요한 가중치 상수인 λ와 관리 한계치를 결정할 때 사용하는 범위 상수 L, 그리고 z의 평균 및 표준 편차를 계산할 때 사용할 z의 범위인 window size(n)를 구해야 한다.

이러한 파라미터 튜닝 작업은 세 개 게임에 대해 각각 진행하였는데, 여기서는 블레이드 앤 소울에서 진행한 데이터를 기준으로 설명하겠다. 블레이드 앤 소울(B&S) 의 경우 2014년 3월 1일 ~ 2014년 5월 13일 (총 74일) 기간의 데이터를 이용하여 유저의 게임 봇 확률을 구했으며 이 데이터를 토대로 적절한 파라미터 값을 정하였다.

먼저 z를 계산하는데 사용하는 가중치 상수인 λ를 결정하기 위해 SSE(Sum of Squared Error)를 사용했다. z는 x에 대한 가중치 이동 평균이다. 우리는 게임 봇 확률을 계산하는 모델의 성능이 일정하게 유지되는지를 예측하기 위해 이 값을 사용하기 때문에 t 시점에 계산한 이동 평균인 z_t 는 그 다음 시점에 계산된 x_{t+1} 값과 최대한 오차가 작게 끔 계산하는 것이 이상적이다. 그래서 실제 관측치인 x_t와 비교해서 가장 작은 오차를 갖는 z_{t-1}을 계산할 수 있는 λ를 정하였다. Fig. 9-(a)를 보면 λ값이 커질수록 초반에는 SSE가 급격히 감소하다가 일정 수준 이상이 되면 다시 증가하는 것을 알 수 있다. 따라서 변곡점이 되는 지점이 가장 SSE를 최소로 하는 λ값이 된다.

다음에는 관리 한계치 계산 시 사용하는 z의 평균 및 표준편차를 계산할 때 얼마나 오랜 기간의 z 값을 사용할지 결정 한다. 기간 n을 0 ~ 74까지 변경해 가면서 z의 평균 및 표준 편차가 어떻게 변하는지 살펴보면 사용하는 데이터가 많아질수록 (즉, 기간이 늘어남에 따라) 점차 값이 수렴하는 것을 관찰할 수 있다(Fig. 9-(b),(c)). 모델의 성능을 안정적으로 모니터링하기 위해선 가급적 변화에 강건한 기준을 잡는 것이 좋다. 따라서 z의 평균 및 표준편차가 충분히 수렴하기 시작하는 기간(여기서는 $n=60$)을 n으로 정한다.

마지막으로 관리 한계치를 결정하는 상수인 L을 결정 한다. 이를 위해 앞서 결정한 $\lambda=0.15$, $n=60$ 을 적용하여 계산한 z의 변화 그래프를 그리고 이 값의 변화폭을 최대한 수용할 수 있는 관리 한계치의 상한선과 하한선을 그린다(Fig. 9-(d)). 우리가 실험한 블레이드 앤 소울의 경우 실험에 사용한 기간인 14일 간에 대해서 z의 변화폭을 수용할 수 있는 최소 정수인 38을 L로 정하였다.

Table 6. EWMA Parameters

Game	λ	n	L
Lineage	0.15	60	20
Aion	0.95	60	4
B&S	0.15	60	38

리니지 및 아이온에 대해서도 위에서 설명한 것과 동일한 방식을 사용하여 각 파라미터를 정할 수 있으며 우리가 실험을 통해 정한 게임별 파라미터는 Table. 6 에 정리했다.

VI. Conclusion

게임 봇을 탐지하는 것은 전형적인 분류 문제이며 따라서 다양한 분류 알고리즘을 이용해서 해결이 가능하다. 때문에 특정 기간에 주어진 데이터를 이용해서 일반 유저와 게임 봇을 구분하는 것은 여러 가지 기법을 조합해서 수행할 수 있다. 그러나 게임 서비스를 운영하는 회사 입장에서는 게임 봇 탐지가 일시적인 조치가 아닌 상시적인 운영이 필요한 업무이기 때문에 다양한 게임에 두루 적용할 수 있으면서 적은 유지 보수 비용 만으로 지속적으로 사용할 수 있어야 한다. 이를 위해 우리는 다양한 게임에서 동일하게 적용 가능한 특질 데이터로 자기 유사도를 추출하는 기법을 제안하고, 현재 국내에서 라이브 서비스 중인 리니지, 아이온, 블레이드 앤 소울을 이용한 실험을 통해 이 알고리즘이 안정적인 성능을 보임을 확인하였다. 또한 탐지 모델을 낮은 비용으로 유지 보수하기 위한 방안으로 EWMA 기반의 탐지 모델 자동 학습 알고리즘을 제안하였다. 이 알고리즘을 이용하면 탐지 모델이 최신 데이터에 적합하지 않게 되는 시점을 감지할 수 있으며 이 시점에 모델을 재학습함으로써 최신 데이터에 맞는 모델 재생성이 가능하다. 우리는 이 알고리즘 역시 라이브 시스템에 구현하였고 이를 통해 지속적으로 모델 모니터링 및 자동 갱신을 진행하고 있다.

현재 이 논문에서 제안한 기법으로 탐지 시스템을 구축하여 리니지는 2015년 2월부터 게임 운영에 활용하고 있으며 아이온과 블레이드 앤 소울은 현재 라이브를 위한 검증 및 테스트를 진행 중에 있다.

VII. Future Works

우리가 제안한 탐지 기법은 게임 봇 유형 중에서 주로 사냥이나 채집 등을 통해 빠른 성장 및 재화 획득을 목적으로 플레이하는 파밍 봇에 초점을 맞추고 있다. 그러나 실제 MMORPG 상에는 이런 파밍 봇 외에도 특별한 목적을 위해 플레이하는 악성 캐릭터들이 존재한다. 대표적인 것이 파밍 봇이 획득한 재화를 전문적으로 취합해서 관리하는 뱅커 봇이다. 비록 우리는 이전 연구[13]에서 이미 그래프 클러스터링을 이용한 뱅커 봇 탐지 기법을 제안한 바 있지만 범용성이나 지속성 측면에서 한계가 있어 실전에서 지속적으로 사용하지 못했다. 우리가 사용한 방법 외에도 이런 뱅커 봇 및 대규모 게임 봇으로 이뤄진 '작업장 그룹'을 탐지하기 위해 연구된 다양한 기법들이 있으며 이들 연구는 공통적으로 네트워크 분석을 기반으로 하고 있다[14],[15],[16][17]. 그러나 기존 연구들은 대부분 제한된 데이터와 앞서 언급한 정답 집합 및 검증 상의 한계로 인해 실효성에 대한 효과적인 검증이 이뤄지지 못했다. 따라서 우리는 작업장 그룹을 효과적으로 탐지하기 위해, 본 논문에서 제안한 기법을 사용해서 얻은 대규모 게임 봇 탐지 정보를 토대로 기존 연구들을 검증하고 개선하여 실전에 적합한 알고리즘 및 프로세스를 개발할 계획이다. 마지막으로 우리가 제안한 EWMA 기반의 탐지 모델 유지 보수 알고리즘의 경우 탐지 알고리즘처럼 정량적인 성능 검증을 하지 못했기 때문에 앞으로 시스템을 운영하면서 그 실효성에 대한 검증을 계속 진행할 필요가 있다.

References

[1] Ahmad, M.A, Keegan, B, Srivastava, J, Williams, D and Contractor, N, "Mining for Gold Farmers: Automatic Detection of Deviant Players in MMOGS," Proceedings of the Computational Science and Engineering International Conference, vol. 4, pp. 340-345, Aug, 2009

[2] Thawonmas, R, Kashifuji, Y and Chen, K, "Detection of MMORPG Bots Based on Behavior Analysis," Proceedings of Advances in Computer Entertainment Technology Conference, pp. 91-94, Dec, 2008

[3] Kang, A, Kim, HK and Woo, J, "Chatting Pattern Based Game BOT Detection: Do They Talk Like Us," Journal of KSII Transactions on Internet and Information Systems, vol. 6, no. 11, pp.2866-2879, Nov, 2012

[4] Kang, A, Woo, J, Park, J and Kim, HK, "Online Game Bot Detection Based on Party-Play Log Analysis," Journal of Computers and Mathematics with Applications, vol. 65, no. 9, pp. 1384-1395, May, 2013

[5] Chung, Y, Park, C, Kim, N, Cho, H, Yoon, T, Lee, H and Lee, J, "Game Bot Detection Approach Based on Behavior Analysis and Consideration of Various Play Styles," Journal of ETRI, vol. 35, no. 6, pp. 1058-1067, Dec, 2013

[6] Chen, K, Liao, A, Pao, H and Chu, H, "Game Bot Detection Based on Avatar Trajectory," Proceedings of Entertainment Computing ICEC 2008, vol. 5309, pp. 94-105, 2009

[7] Kesteren, M, Langevoort, J and Grootjen, F, "A Step in the Right Detection: Bot Detection in MMORPGs using Movement Analysis," Proceedings of The 21th Benelux Conference on Artificial Intelligence, 2009

[8] Crovella, M.E, and Bestavrow, A, "Self-similarity in World Wide Web traffic: evidence and possible causes," Journal of IEEE/ACM Transactions on Networking, vol. 5, no. 6, pp. 835-846, Dec, 1997

[9] Kwon, H, Kim, T, Yu, S and Kim, HK, "Self-similarity bases Based Lightweight Intrusion Detection Method for Cloud Computing," Intelligent Information and Database Systems, Springer Berlin Heidelberg, pp. 353-362, 2011

[10] Kwon, H and Kim, HK, "Self-similarity Based Bot Detection System in MMORPG," Proceedings of The 3rd

International Conference on Internet, 2011
〔11〕 Hunter, J.S, "The Exponentially Weighted Moving Average," Journal of Quality Technology, vol. 18, no. 4, pp. 203-210, 1986
〔12〕 Roy, A, Ahmad, M.A, Sarkar, C, Keegan, B and Srivastava, J, "The Ones That Got Away: False Negative Estimation Based Approaches for Gold Farmer Detection," Proceedings of IEEE PASSAT and SocialCom, pp. 328-337, Sep, 2012
〔13〕 Lee, E, Lee, J and Kim, J, "Detecting the Bank Character in MMORPGs by Analysis of a Clustered Network," Proceedings of The 3rd International Conference on Internet, 2011
〔14〕 Keegan, B, Ahmad, M.A, Williams, D, Srivastava, J and Contractor N, "Dark Gold: Statistical Properties of Clandestine Networks in Massively Multiplayer Online Games," Proceedings of IEEE SocialCom, pp. 201-208, Aug, 2010
〔15〕 Woo, K, Kwon, H, Kim, HC, Kim, C and Kim, HK, "What can Free Money Tell Us on the Virtual Black Market," Proceedings of ACM SIGCOMM Computer Communication Review vol. 41, no. 4, pp. 392-393, Aug, 2011
〔16〕 Kwon, K, Woo, K, Kim, HC, Kim, C and Kim, HK, "Surgical Strike: A Novel Approach to Minimize Collateral Damage to Game Bot Detection," Proceedings of Annual Workshop on Network and Systems Support for Games. pp. 1-2, Dec, 2013
〔17〕 Fujita, A, Itsuki, H and Matsubara, H, "Detecting Real Money Traders in MMORPG by Using Trading Network," Proceedings of the 7th Artificial Intelligence and Interactive Digital Entertainment Conference, 2011
〔18〕 Cevizci, I, Erol, M, Oktug, S.F, "Analysis of Multi-Player Online Game Traffic Based on Self-similarity," Proceedings of the 5th Workshop on Network and System Support for Games, NETGAMES 2006, Oct, 2006

〈저자소개〉

이 은 조 (Eun Jo Lee) 정회원
2002년 2월: 숭실대학교 컴퓨터학부 학사
2007년 2월: 숭실대학교 정보통신대학원 석사
2015년 2월~현재: 고려대학교 정보보호대학원 박사과정
2007년 5월~현재: 엔씨소프트 데이터인텔리전스팀 팀장
〈관심분야〉 온라인게임 보안, 데이터 기반 보안

조 원 준 (Won Jun Jo) 정회원
2002년 2월: 연세대학교 산업시스템공학 학사
2002년~2004년: NHN
2004년~2012년: SK 커뮤니케이션즈
2012년~현재: 엔씨소프트 데이터인텔리전스팀

이 지 나 (Jina Lee) 정회원
2005년 2월: KAIST 산업공학과 학사
2005년 1월~2006년 6월: 삼성전사 경영혁신팀
2009년 2월: KAIST 산업및시스템공학과 석사
2010년 1월~2014년 9월: 엔씨소프트
2014년 10월~현재: 카카오 비즈추천팀

김 현 철 (Hyun Chul Kim) 정회원
2007년 3월~현재: 엔씨소프트 ITM 그룹 TMO 팀

엄 혜 민 (Hye Min Um) 정회원
2009년 12월~현재: 엔씨소프트 데이터인텔리전스팀

권 혁 민 (Hyuk Min Kwon) 학생회원
2009년 2월: 광운대학교 컴퓨터소프트웨어 학사
2012년 8월: 고려대학교 정보보호대학원 석사
2012년 9월~현재: 고려대학교 정보보호대학원 박사과정
2014년 9월~현재: 한국정보통신기술협회 소프트웨어시험인증연구소
〈관심분야〉 악성코드, 온라인게임 보안

김 휘 강 (Huy Kang Kim) 종신회원
1998년 2월: KAIST 산업경영학과 학사
2000년 2월: KAIST 산업공학과 석사
2009년 2월: KAIST 산업및시스템공학과 박사
2004년 5월~2010년 2월: 엔씨소프트 정보보안실장, Technical Director
2010년 3월~2014년 12월: 고려대학교 정보보호대학원 조교수
2015년 1월~현재: 고려대학교 정보보호대학원 부교수
〈관심분야〉 온라인게임 보안, 네트워크 보안, 네트워크 포렌식