AWS FOR GAMES

一緒に学び考える！

ゲームの AI/ML 活用

Sheng Hsia Leng
アマゾン ウェブ サービス ジャパン合同会社
ソリューションアーキテクト

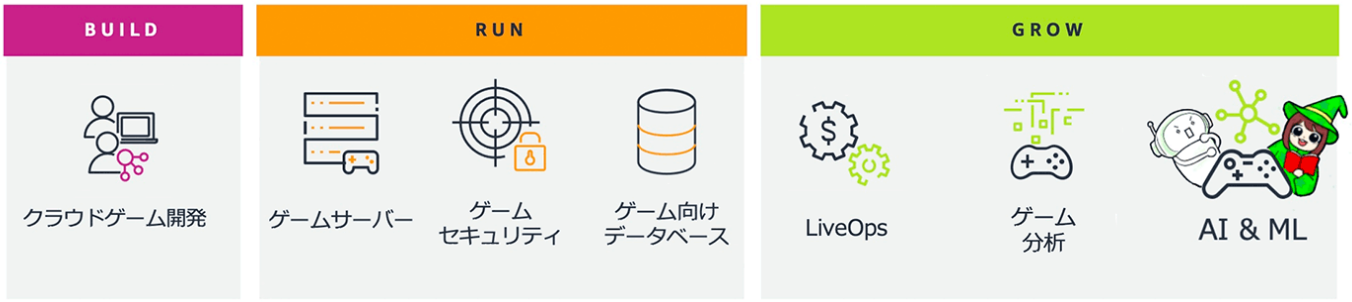
中村　一樹
アマゾン ウェブ サービス ジャパン合同会社
ソリューションアーキテクト

ゲームなみなさんこんにちは、Game Solutions Architect の Leng (@msian.in.japan) と Game Solutions Architect の Kazuki です。

この投稿ではゲームにおける AI/ML の活用を試してみたいが正直何ができるかわからない、と言った方々の疑問や不安を少しでも取り払って AI/ML をゲーム開発や運営に役立てることができるようになることを目指した記事になります。

AWS for Games

AWS for Games ではより早い開発、よりスマートな運営、そしてより楽しいゲームへの成長という **Build**、**Run**、**Grow** の 3 つの柱に沿ってサポートします。本日は **Grow** の柱、 AI&ML の入門の部分になります。

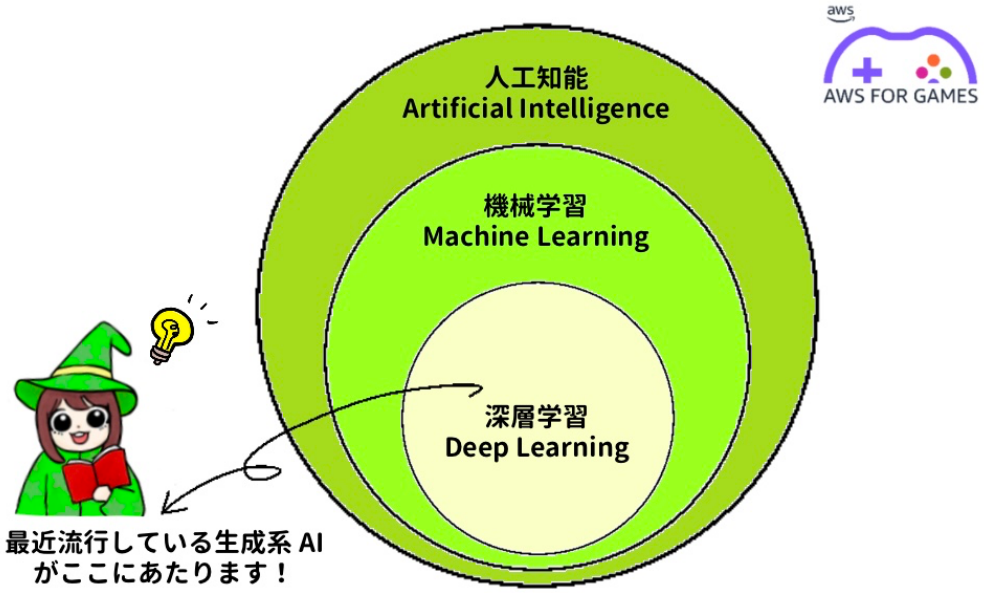


そもそも AI/ML とは

最近は AI という言葉を聞かない日はないくらい日常に AI という言葉が溢れています。しかし AI と聞くと「なんかすごいことをやってくれる何か」「すごく賢いことをする何か」という印象を持たれている方が多いと思います。そこでまずはこの AI が何者なのかを簡単に理解した後に実際にゲーム開発、運営のどのような場面で活躍するのかを探っていこうと思います。
AI は「**Artificial Intelligence**」の略語で、日本語では人工知能と呼ばれています。AI に関しては明確な定義が定まっておらず何ができたら AI で何かできなれば AI ではない、と言ったものはありません。「人間の知能に近い機能を持ち自動的に判断をするもの」といった感じのものです。

ML は「**Machine Learning**」の略語で、日本語では機械学習と呼ばれています。人工知能を支えるの技術の一つとされています。世間一般で聞く AI と呼ばれるもののほとんどは実はこの ML です。モデルの学習方

法の違いに応じて、教師あり学習、教師なし学習、強化学習の 3 つに分類にされます。また分類とは別に ML の中に DL (Deep Learning) と呼ばれるものがあり、こちらは日本語では深層学習と呼ばれるものです。最近流行している自動で何かを作ってくれる 生成系 AI はこの区分になっているものがほとんどです。ML も DL も AI に内包されているものです。そのため一般的には全てひとまとめに AI と呼ばれています。



AI は何ができて何ができないのか

「AI は万能で何でもできる」、「仕事を奪われる」などと言われていますが、決して万能ではありません。ほとんどの AI は教えられたことしか行うことができず、そのためには教える過程や学習が必要になります。学習データと学習方法次第で賢くなることもあれば、賢くならない場合もあります。また、最近は用途に適した学習アルゴリズムが確立されていますが、学習データの前処理に関してはまだまだ自力でやる必要があります。そのため効率よく学習をさせるための学習データを AI エンジニアが作成したい AI の内容に応じて作らなければなりません。AI エンジニアが AI を作成するために行う学習の過程の大半はこの「学習データの準備がほとんど」と言っても過言ではありません。

少し話が逸れてしまいましたが、 AI ができることは学習された内容の範囲内のことしか行うことはできません。0 から 1 を作り出すことはできません。例えば、ゲームを作ることができる AI が作られたとして、シューティングゲームとアクションゲームを学習させた場合、 TPS などの類似したゲームが生まれる可能性はありますが、レースゲームが生まれることはありません。 AI は現状、人が「作業」と言っているもののほとんどと、アイディアの組み合わせによる「創造」の一部を行うことが可能になっていると思われます。

昔からゲームで使われていた AI と今 AI と言われているものの違い

ゲーム開発において、古くから「AI」という言葉は使われてきていました。それは主に NPC (Non Player Character) のことでした。ほとんどのゲームではプレイヤー以外のキャラクターが存在していますが、これらのキャラクターはプログラマーが組み込んだプログラムに基づいて判断し行動しています。ある状況 (データ) をもとにプログラム (学習) によって判断が行われます。これは機械学習ではありませんが、AI の一形態と言えます。

しかし、最近の AI とは 主に機械学習のことを指すため、「ゲーム開発に AI を使う」という表現には、「昔から使われている」という印象を受ける方も少なくありません。また、実際に AI 導入の PoC (Proof of Concept) を行っている際も「AI を使って自動プレイ AI を作成して・・・」といったように AI という単語が乱立して混乱を招くこともあります。

ゲームと AI/ML

AI/ML がどのようなものなのかを簡単に説明させていただきました。ではこの AI/ML はゲーム開発、運営ではどのように利用できるのでしょうか。大きな分類としてはとしてはゲームの外で AI を活用するか、ゲーム自体に AI を組み込むか、という二つに分かれます。

ゲーム外 AI

AI をゲームの外側で利用するものになります。「ゲーム開発で利用する」、「ゲーム運用で利用する」、といったさまざまなものがあります。ゲームの内部に AI 技術を組み込まず、外部システムを利用することがほとんどなのでリソースの制限等なく、AI を導入しやすい分類になります。

ゲーム開発に利用する

アセット自動生成

アセットとは画面に映る 3D モデル、2D イラスト、アニメーションデータ、エフェクト、サウンド、テキスト、ステージデータなどゲーム開発に必要な素材です。

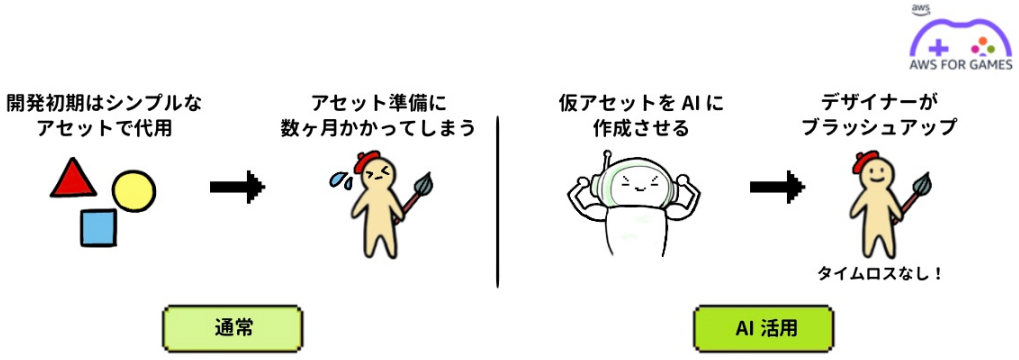
生成系 AI を活用することでゲーム内アセットを自動生成することで開発効率を上げるという活用方法です。

プロトタイプやゲーム開発時の仮アセットの作成

ゲーム開発初期で実際に考えた企画内容が面白いかどうか、を確認するためにプロトタイプを作成することがほとんどです。このプロトタイプ開発は「今後数年間かけて開発するゲームが面白いものなのか」、「売れるものなのか」を判断するとても重要なものです。しかしこのプロトタイプに数年もかけるわけにはいかず、大体 3 ヶ月 ~ 6 ヶ月ほどで作られることがほとんどです。いかに早くこのプロトタイプを作成しゲーム開発を始めることができるか、というものはゲーム開発会社にとって大きな課題となっています。

しかし、そのプロトタイプ作成はすぐに始めることができません。まず開発に必要なアセットを作成する必要があります。製品版相当のアセットが必要なわけではなく、中には誰でも作成できそうな丸や四角などで代用できるものありますが、基本的に面白さが確認できるレベルの質は必要です。そのレベルのアセットができてから初めてエンジニアはそのアセットを使ってゲームを開発していくことができます。このアセット準備に数ヶ月かかってしまってはプロトタイプ完成が遅くなりますが、開発初期はデザイナー数名、プログラマ数名など小規模で行われることが多いのでどうしても時間がかかってしまいます。

この仮アセットの作成は AI を活用することにより大幅にコストを削減できる可能性があります。生成系 AI を使用することで様々なリソースを作成し、そのアセットを利用してプログラマーが開発を始め、デザイナーは仮アセットのブラッシュアップや必要なリソースを作成します。最後に正式リソースに置き換えることでプログラマーは最初からプログラミングが行えロスタイムなく、早くプロトタイプを作成することができます。また、仮アセットを利用するアプローチはプロトタイプ制作だけでなく実際の本開発にも活用することができます。



量産が必要なアセットの作成

ゲーム開発に必要なアセットはどんどん増えていっています。30 年ほど前のゲームではセーブなどなく 1 つのゲームで数時間遊べればよかったものが、今では何十時間、何百時間遊べるものが当たり前になっています。そのため必要なアセット数もどんどん多くなってきています。そのアセット内でもゲームにとって重要なアセットと、必要だが重要ではないアセットがあります。重要なアセットというのは主人公や仲間のデザインや 3D モデル、モーションであったりメインストーリのシナリオなどそのゲームのキモとなるアセットです。

しかし RPG の街に配置されている NPC の他愛のない会話などはゲームを彩る一部になりますが世界観などを壊していなければどのような会話がされていても問題ないです。そのため、主人公などのアセットに比べると重要度は下がりますが、ゲーム内には必要なアセットなので作っていかなければなりません。このような細かなアセットも含めると一つのゲームを完成させるために作らなければいけないアセット量は膨大になってしまいます。

この「必要だが重要度の低いアセット」の作成を AI に行わせることで開発コストを下げることが期待されています。NPC の会話やモブキャラクターのデザインなどを AI に作成させ、そのまま使えるものを使い、難しいものは AI でできたものをベースにブラッシュアップすることでクオリティを落とすことなく開発コストを下げることできます。イメージとしては AI に完成度 60 ~ 70 % のものを作成させ、残りを開発者が行うといったものです。デザイナー系のアセット、シナリオやセリフ、ステージデータ、など量産が必要なアセットに対して効果をが期待されます。

AWS では Amazon Bedrock という 生成系 AI サービスが発表されました。基盤モデルを API 経由で利用でき、専用データによるカスタマイズができます。用意する専用データは数件でもかまいません。文章や画像の生成ができる基盤モデルが利用できるので上記の例の使い方をができるようになり開発効率の向上が期待されます。
※Amazon Bedrock は 4 月 25 日現在 Limited Preview となっているためお使いいただくことはできません。

表記揺れ、誤字、脱字検知

AI を使用することで文章内の誤字や脱字、表記揺れなどを検知し、内容を修正する活用方法です。

キャラクターのセリフやストーリー、チュートリアルなどのテキストの誤字脱字、表記揺れの検知、修正案の提示

ゲームのボリュームがどんどん増えていくにつれて作成しなければならないテキスト量は比例して増えていきます。キャラクターのセリフ、ナレーション、ストーリー解説、チュートリアル、画面 UI などゲーム内容に関わるものから、遊び方などを説明する説明書などゲームをフォローするためのテキストもあります。

この膨大な量のテキストを作成していると、どうしてもキャラ名などの固有名詞の間違い、キャラクターの一人称や口調が変わってしまうといった誤字や脱字、表記揺れが多く発生してしまいます。打ち終わったテキストの確認は行いますが、自分で作成したものの間違いを見つけるのは難しく、他人がチェックしたとしてもすり抜けてしまう可能性もあります。テキスト自体の量も多く、口調などの表記揺れは見つけるためのスキルも高く属人性も高い作業になりがちです。

このチェックは AI を活用することで削減できる可能性があります。AI は特徴や間違いを見つけることが得意なので誤字、脱字、表記揺れをチェックさせ、指摘させます。その後、前後も文脈から正しい文言の候補を出すことにより修正コスト自体も削減することができるかもしれません。

全文検索

AI を使用して膨大な量のデータの中から必要な情報の検索をサポートする活用方法です。

新規メンバー追加時の情報案内

ゲーム開発は数年にわたる長期戦です。そのため開発メンバーが入れ替わる、追加されるということがあります。新規メンバーの一番最初の仕事はゲーム開発環境を整えることです。多くの開発チームでは新規メンバー向けの環境設定やチーム運営などがまとまった資料やリンク集が用意されています。しかし、その資料やリンク集が整備されておらず古い情報だったりすることがあります。普段の開発では使用されず、最初の 1 回行うだけの資料やリンク集のため更新されづらいものになります。そのため新規メンバーのサポートなどにコストがかかってしまいます。

この問題は検索系 AI を活用することで解決できる可能性があります。「やらなければならないもの」、「キーワード」は普遍なのでその情報を AI に渡すことで必要な最新の情報が提案されます。そこから新規メンバーが自分が必要な資料を確認することで新規メンバーのみで環境設定やチームルールなどを確認できるので、他のメンバーのサポートを最小限に抑えることができます。

素早く資料を見つける

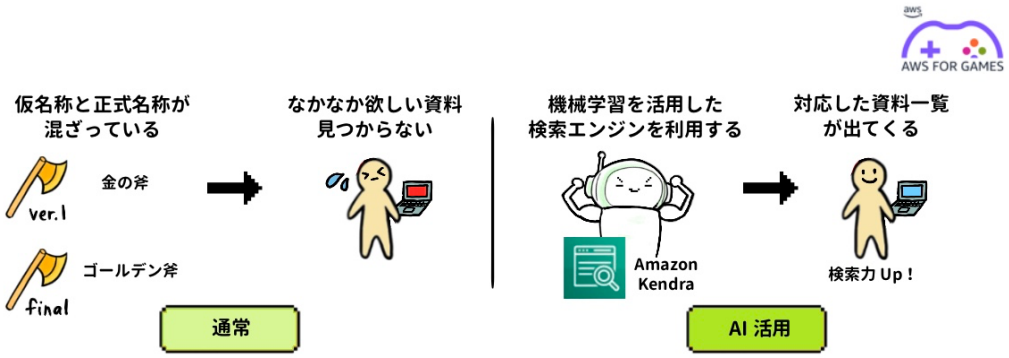
ゲームを開発している過程で多くの資料が作成されます。ゲームの企画書、仕様書、打ち合わせの議事録などゲーム内に組み込まれないが必要なドキュメントなどの情報が多くあります。それらは社内ストレージや情報管理ツールにまとめられています。ツールには簡単な検索機能がありますが、簡単に見つからないものもあります。その他資料もフォルダで管理されていますが、長期開発になると不適切なフォルダに混入したり、適切なフォルダがどれなのかわからない、などで常に整理されているかどうかもわかりません。ゲーム開発において資料を探す、という行為は頻繁に行われますが、議事録や少し古いもの、自分の担当外の資料を確認する、となると見つけるまでに時間がかかってしまいます。この時間はちりつもですが確実に開発の時間を消費しています。

この検索部分を AI に任せることで誰でも簡単に資料を見つけやすくなります。例えば「エネミー A の攻撃に関する資料が欲しい」と思った場合、AI に「エネミー A の攻撃資料」というように命令すると「エネミー A の攻撃に関わっている資料の一覧」が表示されます。ユーザーはその中から自分が欲しかった資料を選択することができます。何がどこにある、といったことを意識せずに欲しい資料を見つけることが容易になるので開発者の資料検索コストを大幅に下げることが期待できます。

正式名称と仮名称の紐付け検索

ゲーム開発の初期ではゲームタイトル、キャラクター名、など、ほぼ全てのものに名前が決まっていないことがが多いです。続編であれば過去作の名前がついていますが新規タイトルの場合は見た目や管理 ID で仮の名前が付けられています (一番最初に作った武器なので wp001 や金色の斧なので金の斧など)。そして開発中盤から後半にかけて名前が決まってくると資料内にも正式名称と仮名称が混在するようになってきます。そうなるとあの資料は正式名称でこの資料は仮名称で、といった形になり検索する側が気をつけないと資料を見つけにくくなってしまいます。後半から参加したメンバーだとそもそも正式名称しかしらないので過去の資料を見つけることが困難であり、古くからいるメンバーは逆に正式名称に慣れていないので仮名称を使ったりしてしまいます。仮名称や管理 ID と正式名称の対応表なども用意されることがほとんどですが、覚えていないと毎回見にいく必要があり手間になります。

この新旧入り混じった資料の検索効率を AI で向上させることが期待されます。管理 ID、仮名称と正式名称が紐付けることができるので正式名称で検索しても仮名称で検索しても欲しい情報を簡単に取得することができます。



ゲーム内マニュアル検索の補助

ゲーム内でわからないことがあった場合はゲーム内マニュアルを確認することが多いと思います。その際にマニュアルがメニュー分けされていても「自分が調べたい情報がどのメニューに入っているのかわからない」といったことも起きると思われます。外部サイトで検索してもらえればまだ良い方ですが、「わからないからもういい」と諦めてしまうと運営側は困ります。

その問題をこの全文検索で解決できるかもしれません。全文検索をゲーム内に組み込むことによってゲーム内でわからないことをユーザー自身で簡単に調べることができるようになり、ゲームへの理解が深まる可能性があります。このことによりゲームの機能などに関する運営側への問い合わせも減らすことが期待できます。

AWS サービスを利用して全文検索を利用する

AWS では Amazon Kendra という全文検索エンジンサービスがあります。これを使うことにより上記ユースケースの問題を解決することが期待できます。新メンバー、既存メンバー関係なくチーム全体の検索力を上げることにより開発効率を向上させることが可能になります。

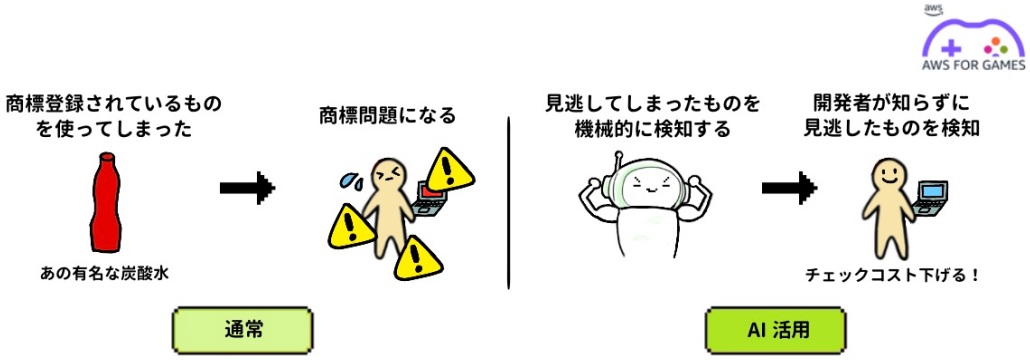
動画、画像検知、アセット検知

AI を活用することである画像や動画内のある特定のオブジェクトを検知し、クオリティ向上に役立てる活用方法です。

仮データを検知し、ゲームのクオリティを上げたり商標問題を解決する

アセット自動生成の項目でも記載しましたが、ゲーム開発では開発初期にまず「仮データ」で開発し、その後正式なデータに置き換えて開発を進めていくことがほとんどです。しかしその仮データを何らかの理由で変え忘れてしまうとゲーム全体のクオリティが下がってしまいます。仮データのクオリティがある程度高くて見逃してしまう、明らかに仮データであることがわかるようにしていたとしてもチェックをすり抜けてしまうこともあるかもしれません。その場合は仮データのままリリースされてしまうことになります。また、仮データを商標登録されているものを使用してしまっていた場合、意図していない問題に波及する可能性があります。商標登録されているものは多いので開発者が気付かずに使っているケースもあるかもしれません。

これらを検知するために AI の活用が期待されます。AI を活用することで人の目では見逃してしまったものを機械的に検知することができ、チェックコストを下げるだけでなく、開発者が知らずに見逃していたものも検知することができます。



画面崩れ、意図していない画面の検知

ゲームには処理を軽くするために遠くのオブジェクトをローポリゴンにする、表示から消してしまう、など様々な工夫がされています。しかしそのような工夫が意図していない条件や不具合により発動してしまい、表示されていないといけないものが表示されていない、近くなのにローポリゴンになっている、ポリゴンが伸びてしまい変な形になる、変な色になる、カメラがステージに当たってしまい画面が激しく揺れる、世界の裏側が見えたりしてしまうものもあります。この不具合は比較的再現は容易ですがある特定の条件や不具合が複合している場合、まず発見することが難しいこともあります。

この意図していない画面の崩れなどを AI に検知させることが期待されています。記事後半に書いてあるエージングテストと組み合わせることにより自動的にゲームをプレイさせ、その時の画面でおかしなことが起きている箇所を検知していきます。このことによりチェックコストを下げることができます。

ゲーム運営に利用する

チートユーザー検知

AI を活用することで何らかの不正行為をおこなっているチートユーザーを検知し、対策を行う活用方法です。

チートユーザーをピックアップする

ランキングイベントや通信対戦で必ず問題になるのがチートユーザーの存在です。このユーザーはゲームアプリに対して何らかの不正行為を行うことによりゲームを有利に進めたり、ゲームバランスを崩壊させたり、普通に遊んでいるユーザーに迷惑をかけるなど、様々なことを行いゲームに悪影響を与え、ゲーム寿命を縮めてしまいます。そのため運営側は対応を行わなければなりません。開発中からこのチートユーザーに対してゲーム内で様々な対応を行っていますが、その対策は破られてしまうことが多く、また対策を行う、といった常にイタチごっこになってしまいます。そのためゲーム内対策もいれつつ、ゲーム外でチートユーザー対策を行なっていきます。ゲームのプレイデータなどの情報からチート行為をおこなっているかの確認を行い、ランキングからの削除や警告、最終的には BAN などで対応していきます。このチェックはユーザーの報告や運営側が人の目で多くのリストから怪しいユーザーをピックアップし、そのユーザーを細かく見ていくといく作業になるので、コストが高い作業になります。またミスも許されない作業なので精神的な負荷も高い作業です。本来は面白いゲームに使われるはずのリソースがチートユーザーにさかれるため著しくゲーム運営の効率が下がります。

この不正な行為やチートユーザーの検知に AI の活躍が期待されています。チートユーザーの特徴を見つけて分類する、おかしな数値や行為を検知する、といった方法でチートユーザーをピックアップすることでコスト

を下げることができます。また、ユーザー報告がないチートユーザーやピックアップ漏れのチートユーザーもチェック対象にできるのでチート検出精度も上がります。ミスが信頼を失うことに繋がるので最終的な判断は運営側の人が行う必要はあると思いますが対応コストは大幅に削減することができます。

またチートユーザーの傾向を分析することでどのようなチート行為が行われているのかを把握することができ、今後の対策に役立てることができます。

予測

AI を活用することで近い未来の数値を予測することで対策などを行う活用方法です。

ゲームサーバーの負荷予測

ゲームサーバーのトラフィックの特徴としてある時間あるタイミングのみ大幅に増える、といったものが多いと思います。アップデート直後、メンテナンス開け、時間限定のイベントなどで 5 分前の数倍、数十倍のトラフィックが一気に流れてくることも珍しくありません。

AWS でも AutoScaling などトラフィックの上昇に応じて自動的にサーバー台数を増やすといった機能はありますが、一気に数十倍のトラフィックが来た場合はサーバー台数の増加が追いつきません。そのため多くのゲーム運営は「これくらい増えるであろう」と予想した台数をあらかじめ増やしておき対応します。この増やす台数は担当者の予測で決めることが多く、アクセスのピーク時にもサーバーが枯渇しないように多めに準備することが多くあります。そのため本来必要ない余分なサーバーを用意することになります。

この予測を AI に行わせることで適量のサーバーを算出することが期待されています。今までの上昇の傾向や現在の DAU などから AI が必要な台数を予測します。その予測の数値を基準として最終的に担当者が決めた数を増設することで余分なサーバーコストを減らすことができます。

ユーザー離脱予測

運営中のゲームにとって「DAU」(Daily Active User) や「MAU」(Monthly Active User) といった 1 日や 1 ヶ月でどのくらいのユーザーが遊んでくれているのか、というものは運営ゲームの存続に関わる大きな数値となります。この数字が下がっていくと運営中のゲームがサービス終了、となってしまう可能性が大いにあります。この数字を下げないようにゲーム運営側は新しいコンテンツを追加したり、コラボしたり、新しいキャラクターを追加したりしています。

しかしどのような対応を行ったとしても残念ながらユーザーは何らかの理由で辞めていってしまいます。ユーザーが辞める理由は様々あると思います。単純にそのゲームに合わなかった、飽きてしまったかもしれません。しかし、もしかしたらゲームが分かりずらい、面白くなるところまで進んでいない、難しすぎる、などという理由で辞めているユーザーがいるかもしれません。後者のユーザーは上記の問題が解決されればそのまま続けてくれるかもしれません。

この辞めるかもしれないユーザーの検知と辞めてしまうユーザーの傾向を知ることを AI では期待されています。辞めそうになっているユーザーを検知できればそのユーザーに対して何らかの対応を行うことができるかもしれません。また傾向を知ることができれば、ゲーム内の導線を変更するなどの対策をゲーム内に組み込むことができます。

感情分析、分類分け

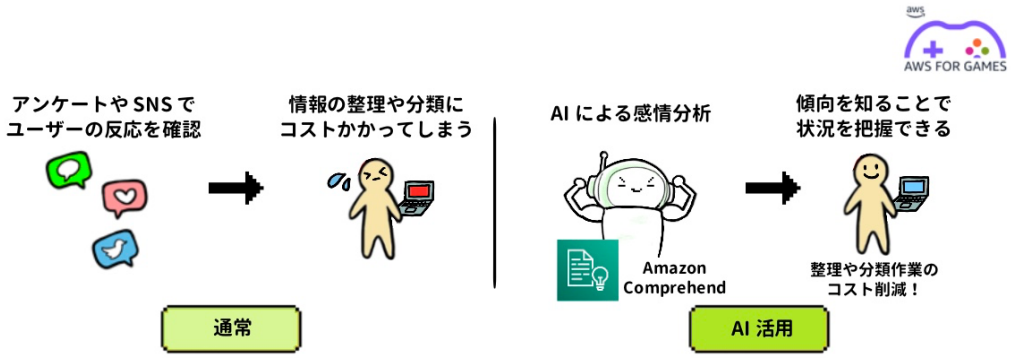
AI を使用して文章の感情分析を行ったりカテゴリごとに分類して分析などに役立てる活用方法です。

アンケートや SNS の文章を分析する

運営中のゲームはユーザーがゲームを楽しんでくれているか、満足しているかを常に気にしています。そのためにユーザーに対して定期的にアンケートを取ったり、SNS で情報を収集してユーザーの反応を確認しています。

しかし大量にとったアンケートや SNS の情報を整理したり分類し、内容を把握することには大きなコストがかかってしまいます。重要な情報源のためこの作業内容を削減することはできません。

この整理や分類の作業を AI に行わせることでコストを削減することが期待できます。
プレイヤーのランク（ライトユーザー、ヘビーユーザーなど）や課金額などのユーザー情報、SNS に書いている内容などでユーザーを分類します。また、ユーザーが好意的な意見を出しているのか、否定的な意見を出しているのかを把握するため感情分析をすることもできます。感情分析を使用することで、その文章がポジティブなのかネガティブなのかを知ることができます。残念ながら確実にポジティブ、ネガティブを判断することは難しいですが、おおよその傾向を知ることはできるので状況把握する程度であれば問題ありません。またこれらで得られた情報を可視化することで分析しやすくなり、有効利用しやすくなります。



AWS サービスを利用して感情分析を行う

AWS では Amazon Comprehend という自然言語処理 (NLP) サービスがあります。このサービスを利用すると文章がポジティブな内容なのか、ネガティブな内容なのかを知ることができます。この情報を元のアンケートデータなどに紐づけ、Amazon QuickSight などの BI ツールで可視化するとアンケートなどの分析が容易になります。

コミュニケーションでの有害検知

AI を活用して有害なコミュニケーションや不正なコンテンツを検知する活用方法です。

文字でのチャットやボイスチャットなどゲーム内にコミュニケーションが取れるツールが入っているものは多くあります。最近では自身の書いたイラストや写真などのユーザー自身が作成したコンテンツ (UGC : User Generated Content) をゲーム内にアップロードできるものもあります。

コミュニケーションツールは正しく使用されれば友人と一緒に遊ぶ、新しいコミュニティを作ると言った、ゲームをより楽しめる環境作りやゲームの継続率を上げることが期待されています。しかし、不正な使い方をされると不快な思いでゲームが楽しめなくなる、犯罪に利用されてしまう、といったことなども起きてしまいます。そのためゲーム運営側は健全なコミュニケーションが行われているかを監視する必要があります。不適切な画像がアップロードされていないか、個人情報などのやりとりがされていないか、暴言がないか、何か犯罪に関係するやりとりが行われていないか、など確認しなければならない項目は多々あります。あらゆるコミュニケーションを一度運営側がチェックをして問題ないものだけアップロードや反映を行うこともできますが、その場合レスポンスが落ちてしまいます。また 24 時間 365 日チェックするということになるので運営側の人の精神的負荷やコストが高いので現実的ではありません。そのため一度アップロードや反映を行い、後からチェックし削除する、といった対応になることがほとんどです。

このコミュニケーションの監視に対して AI の活躍が期待されます。暴言や差別用語、個人情報などの不適切な文言などを検知すると前後の文脈から本当に不適切かを判断しその箇所をマスクする。UGC に関しては不適切なイラストなどを検知するとアップロードできなくする。怪しいコミュニケーションが行われている場合は、これまでのやりとりから判断し検知する、といったことで、健全なコミュニケーションツールを維持しやすくなります。

問い合わせ対応

ゲームで何か不具合があったり、わからないことがあるとユーザーはゲーム運営元に問い合わせを行います。その数は 1 日に数百、大規模な不具合が発生した場合には数千、数万といった問い合わせがくることもあります。その問い合わせを担当者はすべて対応していかなければなりません。しかし問い合わせの中にはマニュアルを見れば解決するものやよくある問い合わせなどを確認すれば解決するものも多いと思われます。自身で解決できる問い合わせを減らし、全体の問い合わせ件数を減らすということはとても重要なことになります。

AI によって実際に担当者が対応しなければならない件数を減らすことが期待されます。問い合わせの前に AI によるチャットボットなどを置くことで、まずは AI に問い合わせ内容を解決させます。この AI にはマニュアルやよくある問い合わせの内容が全て入っているのでその内容で解決できるものは全て AI で対応できます。この AI で対応できないものだけを実際の問い合わせにすることで担当者の負荷を下げ、問い合わせ対応コストも削減することができます。

ゲーム内 AI

AI をゲームの内部に組み込み利用するものになります。実際に遊びに利用する、開発をサポートに必要なオートプレイ機能の組み込み、などの方法があります。実際にゲーム内に組み込みを行うためハードの制限を受けることが多く、簡単に AI の更新を行うことができないのでゲーム外 AI に比べると比較的実装難度が高い傾向にあります。

ゲーム AI と呼ばれているもの

大きく分けると 3 つに分類されます。

- **キャラクター AI**
  ゲーム内の環境、状況を認識し、それをもとに意思決定し自律的に行動する AI です。ひたすらプレイヤーを追いかけるといった単純なものから、あらゆる状況に対応した複雑なものまで多々あります。if 文での分岐判断などプログラミングで作られることが主流でしたが、最近では 機械学習のような複雑な処理を使用したものもでてきています。
- **ナビゲーション AI**
  ゲーム内の地形を認識し、経路をナビゲーションする AI です。障害物のない 2D ゲームなどであればプレイヤーの位置さえわかっていればプレイヤーに近づくことができましたが、障害物や 3D フィールドの段差などがある場合は引っかかったりするのでナビゲーションが必要です。確実に移動できるポイントを複数設定し、そのポイント同士を結びメッシュ構造のような道筋をつくり、状況に合わせてその道筋のどこを通れば良いかを判断します。あらかじめクリエイターによって移動可能ポイントを設置し、ポイント同士を接続してナビゲーションメッシュを作成していることが多いですが、生成系 AI を活用し、ナビゲーションメッシュデータを自動作成し、そこにナビゲーション AI で選択させれば、ナビゲーションメッシュ作成コストを下げることもできるかもしれません。
- **メタ AI**
  ゲーム全体を統括する AI です。プレイヤーの動きや現在のゲーム環境から、敵キャラクターの出現やアイテム量の変化、進行管理などのゲーム全体の状況を俯瞰的な視点で把握してコントロールします。自動的にゲームのバランスなどを調整してくれる AI です。

遊びに利用する

AI を中心とした遊びをゲーム内に組み込む活用方法です。

自分のコピーを作って遊ぶ、他のプレイヤーのコピーと遊ぶ、AI を育てて遊ぶ

対戦ゲームにおいて、あのプロプレイヤーと対戦したい、ランカーと戦いたい、いつでも友人と対戦したい、といった欲求はあると思います。しかしランダムマッチにおいてプロプレイヤーやランカーと対戦できる機会はほとんどなく、友人もプレイヤーが遊びたいタイミングでいつも遊んでいるとは限りません。

この欲求を 100% 叶えることは不可能ですが、AI を活用しプレイヤーのコピーを作り、そのコピーと対戦することで少しは欲求を満たすことができる可能性があります。またコピー同士を対戦させて誰の AI が強いかを競い合う新たな遊びも生まれるかもしれません。また、自分のコピーと戦うことで自分の癖などに気付きやすくなりプレイヤーが上達するかもしれません。コピー AI を作るということで様々な新しい遊びや体験をユーザーに提供できる可能性があります。

また、ユーザーがアクションを起こしたり指示を与えることで AI を学習させて遊ぶゲームも存在しています。学習済み AI を使う、ではなく学習自体を遊びにするゲームが今後増えてくるかもしれません。

しかしこの分野は AI の学習自体をゲーム内で行うためハイスペックなハードを要求してしまいます。また学習部分のみを切り出し外部サーバーなどで対応する場合も学習後の AI をダウンロードしてゲームで動くようにしなければならないのでレスポンスも遅く、すぐにユーザーが楽しめるレベルにするのはとても難しい状況です。

学習済み AI で遊ぶ

(この内容は筆者が考えたアイディアの一例になります)
AI は学習させた内容に対して指示をすることで様々なリソースを作成します。その作成したリソースの中には想定外のものが出来上がることもあります。その想定外を利用して楽しむゲームなども現れてきています。また賢い AI は特定の条件下では人間と区別がつかないこともあります。そこで AI を当てる人狼ゲームのようなものも作ることができるかもしれません。

AI の特性、例えば「人に似ている」「特徴を見つけることが得意」「指示した内容のものを生成する」といったものを活用した新しいゲームがこれからどんどん出てくるかもしれません。

AI に教えてもらう (コーチング)

プレイヤーの上達はゲームの継続率に大きく関わってきます。対戦ゲームで上達せず、ずっと負け続けるという体験はユーザーにとって耐え難い苦痛になります。プレイヤーが練習しても負け続けるとゲーム離れを引き起こしかねません。多くの場合、上達の仕方、正しい練習方法がわからないことが要因となっています。正しく練習すれば一定のレベルにまで上達し負け続けるといった体験をなくすことはできると思います。

AI にコーチングをさせることでプレイヤーの上達を促すことが期待されています。AI がその人の癖などを把握し、その人に合った練習プランを提示することでプレイヤーは自分の弱点を知ることができるだけでなく、スムーズに上達していくことができるかもしれません。

AI にオススメパーティーを教えてもらう（レコメンデーション）

長年運営されているゲームはユーザーが持っているリソース（武器やキャラクターなど）がとても多く、ユーザー自身も管理しきれていない状態になっていることもあります。ほとんどのゲームでこのリソースはパーティー編成や装備編成などで複数使用し、組み合わせて使われます。その状態で新しいリソースを手に入れて、新しいパーティーなどを組もうとした際に自分が持っているリソースでどの組み合わせが最適なのかを考えると思います。パーティー編成を試行錯誤する楽しみがありますが、あまりのリソースの多さのため、パーティー編成はかなり難度が高い作業になってしまいます。

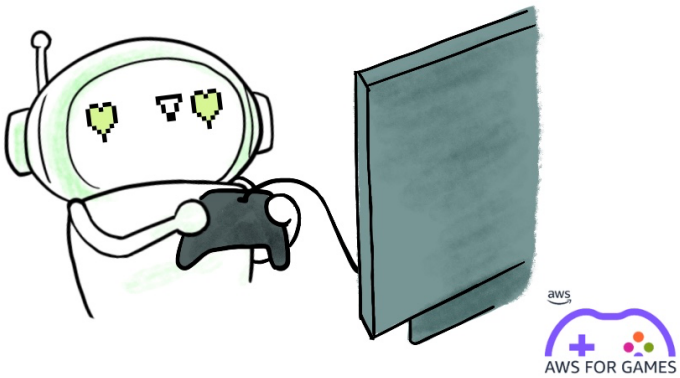
ゲーム内にオススメ編成といった機能があるものもありますが、攻撃力が一番高い組み合わせ、などある数値が最大になるような組み合わせを提示されることが多く、ユーザーが求めているものとは異なる場合があります。そこで外部情報サイトなどを参照して同じものを持っていたらそのままに、持っていない場合は似た性能のものに置き換えなどをして編成をしていますが、置き換えもリソースを把握していないといけないので困難です。リソースが増えることによる編成難度の向上はユーザー離脱のリスクでもあります。

このパーティ編成の補助に AI の活用が期待されます。AI により今のユーザーのリソースの中から最適な組み合わせを提案させることができ、ユーザーはそれをベースにカスタマイズすることで編成の難度を下げることができます。カスタマイズという編成の楽しみを残しつつ、最適なパーティーを編成することができるのでユーザーの負荷を下げつつゲーム側が提供したい楽しみも提供することができます。

開発サポートのための AI 組み込み

オートプレイ AI

AI 自身に自動でゲームをプレイさせることで様々なサポートを行わせる活用方法です。



エージングテストに活用しエージング時間を伸ばす

ゲーム開発の開発終了間際になると、軽微な不具合は修正せず、ゲームが進行不能になってしまうような重大な不具合のみ修正し、基本的に正常に動くかどうかの最終確認をひたすら行うエージングテストを行います。このエージングテストでは製品に手を加えずどれだけ正常に動作していたか、というエージング時間がとても重要になってきます。このエージング時間は同じバージョンでのプレイ時間の合計になるので多くの人が同時に長くプレイする必要があります。開発者やデバッグのメンバーもプレイしますが、致命的な不具合が発生し、修正してしまうと時間はリセットされてしまいます。そのため開発期限の直前で修正してしまった場合には十分なエージング時間を確保することができなくなってしまいます。いかに実時間を短く、エージング時間を伸ばすか、という工夫が必要になってきます。

このエージングプレイを AI に任せることでゲームが動作する環境さえあればエージング時間を短時間で伸ばすことや、人的コストを大幅に下げることが期待できます。今までは人がいなければエージング時間を伸ばすことはできませんでしたが AI を利用することでゲームが動作する環境だけプレイ時間を伸ばすことができます。

早期バグチェックに利用する

ゲーム開発が中盤に差し掛かってくると新規開発が少なくなっていき不具合修正も並行で行っていくことになります。中盤以降はいかに素早く不具合を見つけ、修正していくかが重要になってきます。特に開発終了間際になると、軽微な不具合は修正できなくなります。最近では 1 日目にパッチを当てて修正する、ということも多いですが、それでもパッチを当てる前の内容には不具合が残り続けてしまいます。新規開発中はなかなかチェックに時間を割くことが難しいため不具合の発見はバグチェッカーなどにまかせてしまい発見が遅れてしまうということもあります。早期に発見していればすぐに修正できたものが、後で見つかってしまったため修正難度が上がってしまう、といったこともあります。

この早期発見のためのテストプレイを AI に任せることでバグチェッカーの方の手を煩わせることもなく不具合を見つけることができるかもしれません。自分の開発部分のチェックをオートプレイ AI に任せ、動作に問題がないかを時々確認します。動作が止まっていた場合、すぐに調査を始めることができます。このことによりバグチェックコストを下げることができます。

単純作業のデバッグ作業に AI を使用する

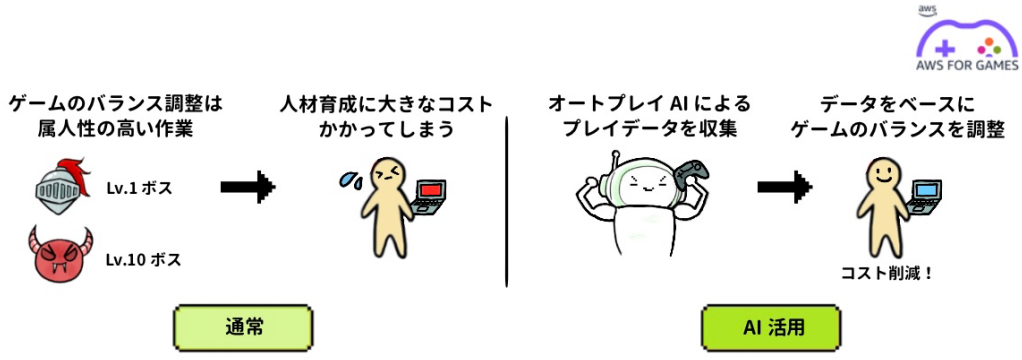
不具合を見つけるためのデバッグ作業の中には単純なものや機械的なものも多く存在しています。全ての選択肢が問題なく進行するか、ひたすら壁にぶつかって壁抜けするポイントはないか、など簡単ではありますがこれらのチェックはとても重要なものです。

このようなチェックに AI を活用することでデバックコストを下げることが期待されます。オートプレイ AI によりこの単純作業を行わせて、定期的に不具合が起きていないかを確認することで、デバッカーは人でないと難しいチェックに専念することができます。

テストプレイに活用しバランス調整サポートを行う

ゲームのバランス調整はかなり属人性の高い作業です。そのゲームに熟知している人が調整するのと新しくチームにジョインした人とでは、かかる時間もクオリィーにも雲泥の差が生まれます。そのため熟知している人しかできない作業になりがちでその人への負荷がとても多くなってしまいます。この属人性を下げるために新しい人を育成するにも大きなコストがかかってしまいます。

このバランス調整を AI にサポートしてもらうことで属人性を軽減させ、コストを削減させることが期待できます。オートプレイ AI にゲームをプレイさせデータを収集し、そのデータを可視化することで難易度などの客観的なデータを見ることができます。可視化されていることによりデータ自体が見やすくなっているので気付きを得る部分も多くなります。この客観的なデータをベースに調整をすることで熟練者と初心者の差を少なくすることができる可能性があります。



まとめ

すでに簡単に試すことができるものから、まだ実現が難しいものまでゲームに活用できそうな AI について私の経験から一例を挙げていきました。この例の他にもまだまだ活用できる分野はあると思います。

今回ご紹介した AI/ML 活用例などが皆さんの今後の AI 活用やゲーム開発、運営のサポートになれば幸いです。