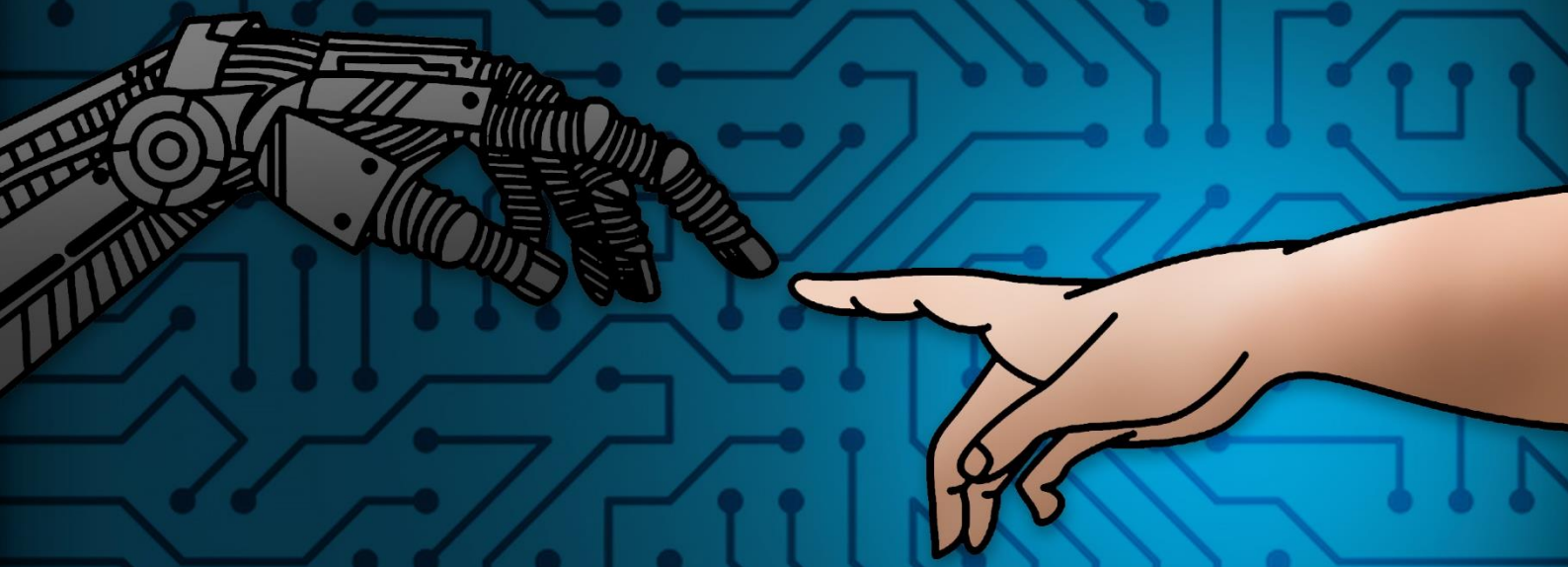
# PROGRAMANT EL FUTUR

## LA REVOLUCIÓ DELS AGENTS INTEL·LIGENTS I DE LES INTEL·LIGÈNCIES ARTIFICIALS



**TREBALL DE RECERCA**

CACAUET

CURS 2023-2024

2N DE BATXILLERAT

INSTITUT PUIG CASTELLAR

ABRIL 2024

“With AI, we will have, for the first time, something that is smarter than the smartest human. There will come a point where no job is needed”

Elon Musk (2023)

**ABSTRACT**

This paper investigates the possibility of developing a fully autonomous virtual agent capable of learning and making decisions independently in the dynamic environment of the Minecraft video game.

The main research question explored is whether it is currently viable, using state-of-the-art artificial intelligence techniques, to create intelligent agents that can learn and act completely autonomously in complex simulated worlds like Minecraft.

The hypothesis posed is that it is possible to develop a virtual autonomous agent that can learn and make decisions in a completely independent way in a dynamic environment such as Minecraft.

To validate this hypothesis, an agent called CraftyAI was implemented using deep reinforcement learning and large language models like GPT-4. CraftyAI was able to perform basic tasks like collecting resources and crafting tools in simplified Minecraft environments.

When evaluated in complex open worlds, CraftyAI exhibited autonomous behaviours like exploration, resource gathering, crafting, and construction. The agent displayed curiosity and continuous learning in the dynamic environment.

The results obtained validate that creating competent autonomous agents in simulated domains like video games is technically viable with current AI methods. However, challenges remain in achieving unbounded progress and curiosity.

This research opens up intriguing possibilities for the future development of increasingly capable AI agents. But it also highlights the need for ethical frameworks as autonomous systems become more prevalent.

**RESUMEN**

Este trabajo investiga la posibilidad de desarrollar un agente virtual completamente autónomo capaz de aprender y tomar decisiones de forma independiente en el entorno dinámico del videojuego Minecraft.

La principal pregunta de investigación explorada es si es viable actualmente, utilizando las técnicas más innovadoras de inteligencia artificial, crear agentes inteligentes que puedan aprender y actuar con total autonomía en mundos simulados complejos como Minecraft.

La hipótesis planteada es que es posible desarrollar un agente virtual autónomo que pueda aprender y tomar decisiones de forma completamente independiente en un entorno dinámico como el de Minecraft.

Para validar esta hipótesis, se ha implementado un agente llamado CraftyAI utilizando aprendizaje por refuerzo profundo y grandes modelos de lenguaje como GPT-4. CraftyAI ha sido capaz de realizar tareas básicas como recolectar recursos y elaborar herramientas en entornos simplificados de Minecraft.

Cuando fue evaluado en mundos abiertos complejos, CraftyAI exhibió comportamientos autónomos como exploración, recolección de recursos, elaboración y construcción. El agente mostró curiosidad y aprendizaje continuo en el entorno dinámico.

Los resultados obtenidos validan que producir agentes autónomos competentes en dominios simulados como los videojuegos es técnicamente viable con los métodos actuales de IA. Sin embargo, quedan retos por superar para lograr un progreso y curiosidad ilimitados.

Esta investigación abre intrigantes posibilidades para el futuro desarrollo de agentes de IA cada vez más capaces. Pero también resalta la necesidad de marcos éticos a medida que los sistemas autónomos se vuelven más prevalentes.

# Índex

## 1. INTRODUCCIÓ

La intel·ligència artificial (IA) s'ha desenvolupat exponencialment en les últimes dècades, demostrant ser capaç d'igualar i fins i tot superar les capacitats humanes en tasques molt concretes com el reconeixement d'imatges, la traducció automàtica o els motors de recomanació. No obstant això, la immensa majoria dels sistemes d'intel·ligències artificials existents en l'actualitat depenen completament de la supervisió, els conjunts de dades d'entrenament i la intervenció humana constant per funcionar correctament.

Els sistemes d'IA reals autònoms, és a dir, capaços de monitorar el seu entorn, establir objectius propis, dissenyar plans d'acció i aprendre sense supervisió, continuen sent un repte obert que la majoria d'experts creuen lluny d'assolir-se. Els principals obstacles són dotar els agents d'un veritable sentit comú, capacitat de raonament i planificació a llarg termini.

No obstant això, en l'última dècada la recerca en camps com l'aprenentatge profund i l'aprenentatge automàtic per reforç ha obert la porta a sistemes que exhibeixen un cert grau d'autonomia per tomar decisions en entorns acotats. És pertinent, doncs, investigar fins on poden arribar aquests avenços en els pròxims anys i explorar si és viable assolir agents virtuals realment autònoms en un futur pròxim, almenys en dominis específics.

Aquest treball explora la possibilitat, que a ulls de molts experts encara sembla remotament futurista, que la intel·ligència artificial pugui prendre decisions completament autònomes, sense requerir cap mena d'intervenció o supervisió humana. L'objectiu principal és determinar tant la viabilitat tècnica com social que la intel·ligència artificial arribi a ser autònoma en un futur pròxim, almenys en dominis específics.

La metodologia emprada constarà de recerca bibliogràfica sobre els fonaments de la intel·ligència artificial, disseny de l'arquitectura i algoritmes de l'agent basant-se en tècniques d'aprenentatge profund, implementació computacional utilitzant frameworks

com PyTorch o TensorFlow, prova del prototip en l'entorn de Minecraft i anàlisi dels resultats.

La meva motivació per al treball ve de la meva fascinació pels ordinadors i la intel·ligència artificial de ben petit. Sempre m'han intrigat aquestes màquines capaces d'executar tasques que semblaven impossibles fa només unes dècades. Passava hores davant l'ordinador jugant i explorant tot el que era capaç de fer. Més endavant, vaig començar a interessar-me per la programació i em meravellava pensant que podia crear els meus propis programes i fer que l'ordinador fes coses útils seguint les meves instruccions.

Amb aquest treball es pretén mostrar la viabilitat de crear agents virtuals autònoms capaços d'operar en entorns dinàmics com els videojocs, fent servir les tècniques més innovadores de la intel·ligència artificial. Els resultats permetran validar els coneixements adquirits i obrir noves línies de continuïtat per a la recerca en aquest apassionant camp.

## 2. HIPÒTESIS

La hipòtesi que plantejaré en aquest treball és la següent:

**És possible desenvolupar un agent virtual autònom que sigui capaç d'aprendre i prendre decisions de manera completament independent, en un entorn dinàmic com el videojoc Minecraft.**

L'agent haurà de ser capaç d'explorar l'entorn, recollir recursos, construir estructures i eines i interactuar amb altres elements del joc de forma autònoma, sense necessitar cap mena de supervisió o guia humana.

Per validar o refutar aquesta hipòtesi, es dissenyarà i s'implementarà un agent basat en algoritmes d'aprenentatge profund i reforç que li permetin desenvolupar les habilitats necessàries per sobreviure i prosperar al món de Minecraft. L'agent haurà de ser capaç d'aprendre a partir de l'experiència, sense dependre de conjunts de dades o intervencions humanes.

Els objectius específics per confirmar la hipòtesi seran:

- Dissenyar una arquitectura d'agent intel·ligent basada en xarxes neuronals profundes capaç d'actuar de manera autònoma.
- Implementar l'agent fent servir llenguatges de programació i frameworks d'aprenentatge profund com PyTorch o TensorFlow.
- Entrenar l'agent en un entorn simulat de Minecraft fent servir algoritmes de reforç.
- Avaluar el comportament de l'agent en diversos escenaris del joc i analitzar la seva capacitat d'aprendre i prendre decisions de forma autònoma.

Si l'agent aconsegueix jugar a Minecraft de manera competent sense supervisió humana, la hipòtesi quedarà confirmada. Els resultats permetran validar que és viable crear agents virtuals autònoms mitjançant les tècniques més innovadores de la intel·ligència artificial.

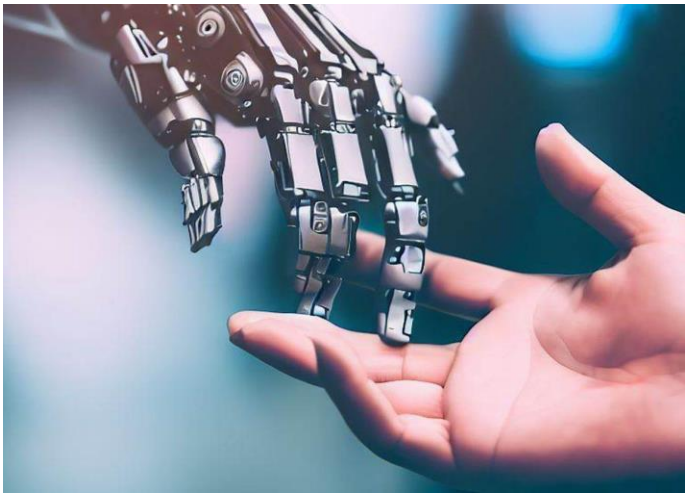
## 3. LA INTEL·LIGÈNCIA ARTIFICIAL

La intel·ligència artificial (IA) és un camp interdisciplinari de la informàtica i les ciències cognitives que té com a objectiu replicar en màquines les habilitats cognitives dels éssers humans. La IA engloba un conjunt de teories, tècniques i tecnologies que permeten que els ordinadors i robots exhibeixin comportaments aparentment intel·ligents similars als dels humans.

Si bé el terme "intel·ligència artificial" va ser encunyat oficialment l'any 1956 durant la Conferència de Dartmouth pels pioners John McCarthy, Marvin Minsky, Claude Shannon i Nathaniel Rochester, la recerca en aquest camp es remunta a principis del segle XX. Alan Turing ja es plantejava el 1950 si les màquines podien pensar mitjançant el seu famós "Test de Turing".

No hi ha una definició universalment acceptada del que és la intel·ligència artificial. Segons l'Associació per a l'Avenç de la intel·ligència artificial (AAAI), la IA es defineix de la manera següent:

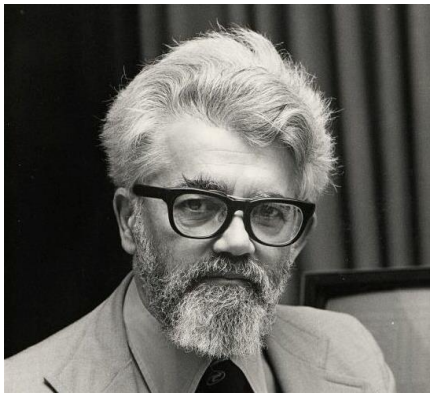
"La intel·ligència artificial refereix a un sistema computacional capaç de dur a terme tasques que normalment requereixen intel·ligència humana, com la percepció visual, el reconeixement de la parla, la presa de decisions i la traducció entre idiomes."



**Imatge 1.** What is AI? *Imatge extreta de https://www.zdnet.com/article/what-is-ai-heres-everything-you-need-to-know-about-artificial-intelligence/*

John McCarthy, considerat el pare de la IA, la va definir originalment com "la ciència i enginyeria de fer màquines intel·ligents". D'altres definicions inclouen:

- “La IA és l'estudi de les facultats mentals a través del processament de símbols” (Marvin Minsky)
- “La IA tracta de construir màquines que puguin fer tasques que requereixin intel·ligència si les dugués a terme un ésser humà” (Stuart Russell i Peter Norvig).



**Imatge 2.** John McCarthy
*Imatge extreta de https://www.wired.com/2011/10/john-mccarthy-father-of-ai-and-lisp-dies-at-84/*

Independentment de la definició concreta, la intel·ligència artificial es caracteritza per:

- Automatitzar funcions cognitives associades a la ment humana com el raonament, la percepció, l'aprenentatge i la resolució de problemes.
- Processar grans volums de dades i informació de manera similar al cervell humà.
- Exhibir comportaments que en humans es considerarien intel·ligents: creativitat, planificació, comunicació, etc.
- Millorar les seves capacitats amb l'experiència i adaptar-se a nous entorns.
- Interactuar amb humans emprant el llenguatge natural
- Dur a terme tasques i prendre decisions de forma autònoma sense intervenció humana.

### 3.1. ORÍGENS I HISTÒRIA

**Imatge 3.** Alan Turing
*Imatge extreta de https://ca.wikipedia.org/wiki/Alan_Turing*

Els primers passos de la intel·ligència artificial es remunten als anys quaranta del segle XX, quan el matemàtic britànic Alan Turing va proposar el que avui es coneix com el Test de Turing, una prova per determinar si una màquina podia exhibir un comportament equivalent a la intel·ligència humana. Turing va defensar que si una màquina era capaç d'enganar a un humà fent-li creure que estava parlant amb un altre humà, aleshores podíem dir que aquesta màquina era intel·ligent.

En aquella època també destaquen les contribucions dels científics Warren McCulloch i Walter Pitts, que van desenvolupar models matemàtics simplificats de les xarxes neuronals biològiques. Aquest treball va asseure les bases per a l'estudi de les xarxes neuronals artificials.

Ja en la dècada dels cinquanta, el matemàtic John McCarthy va encunyar el terme “intel·ligència artificial” en organitzar la conferència de Dartmouth el 1956, considerat el punt de partida d'aquesta disciplina. En aquest esdeveniment es van reunir importants estudiosos en informàtica i ciències cognitives per discutir la possibilitat de crear programes que poguessin imitar comportaments intel·ligents.

Un dels assistents, Marvin Minsky, es convertiria en un dels pioners de la IA en les dècades següents, explorant qüestions com la representació del coneixement i desenvolupant idees sobre agents intel·ligents. Un altre assistent destacat, John McCarthy, va crear el llenguatge de programació Lisp, que seria amplament utilitzat en IA durant molts anys.

Ja en la dècada dels seixanta van sorgir els primers programes capaços de demostrar teoremes matemàtics, com el Logic Theorist desenvolupat per Allen Newell, Herb Simon i Cliff Shaw. Un altre programa destacat d'aquesta època va ser Eliza, creat per Joseph Weizenbaum, que simulava mantenir una conversa amb un terapeuta.

En els anys setanta, els investigadors van començar a aplicar les tècniques de la IA per desenvolupar els anomenats sistemes experts, programes capaços de raonar i prendre decisions en dominis específics igual que ho faria un expert humà. Un dels més reeixits va ser MYCIN, dissenyat per ajudar metges en el diagnòstic de malalties infeccioses.

La dècada dels vuitanta va estar marcada pel desenvolupament de màquines capaces de processar grans quantitats de dades i aplicar tècniques estadístiques i algorítmiques per trobar patrons. Aquest nou enfocament es coneixeria com a aprenentatge automàtic.

Ja en la dècada dels noranta, la IA va fer grans avanços en àrees com el reconeixement del llenguatge natural i la visió per computador. Eines com el reconeixement òptic de caràcters es van generalitzar en aplicacions quotidianes.

Una fita destacada va ser quan el 1997 l'ordinador Deep Blue d'IBM va derrotar al campió mundial d'escacs Gary Kaspàrov. Aquest triomf va demostrar com les capacitats analítiques de les màquines podien superar la intel·ligència humana en dominis específics com els jocs d'estratègia.


**Imatge 4:** Gary Kaspàrov vs Deep Blue *Imatge extreta de https://hipertextual.com/2016/02/deep-blue*

Amb l'arribada del segle XXI, tècniques com les xarxes neuronals profundes van començar a revolucionar camps com el processament de llenguatge natural i el reconeixement d'imatges. Sistemes d'IA van passar a integrar-se en serveis i dispositius utilitzats per milions de persones.

**Imatge 5:** AlphaGo vs Lee Sedol *Imatge extreta de https://www.youtube.com/watch?v=rOL6QJdAlm8*

El 2016, l'ordinador AlphaGo de Google DeepMind va derrotar al campió mundial de Go, Lee Sedol, en una fita històrica, ja que aquest joc d'estratègia era considerat més complex que els escacs. Pocs anys després, el 2018, una nova versió del sistema AlphaZero va aconseguir dominar els escacs sense cap entrenament previ, només jugant contra si mateixa.

En els darrers anys, la IA ha continuat evolucionant ràpidament i s'ha popularitzat l'ús d'assistents virtuals, vehicles autònoms, sistemes de recomanació i altres aplicacions amb capacitats “intel·ligents”.

L'any 2022 va marcar un punt d'inflexió amb l'aparició de models generatius de llenguatge entrenats amb enormes quantitats de dades, com ChatGPT. Aquest xatbot conversacional va causar sensació per la seva capacitat per mantenir converses coherents i generar textos amb un estil proper a l'humà. En pocs mesos va obtenir més de 100 milions d'usuaris.

També han sorgit nous models generatius capaços de crear imatges a partir de descripcions de text, com DALL-E 2, Stable Diffusion i Midjourney. Les imatges generades per aquests sistemes han arribat a confondre's amb fotografies reals.


**Imatge 6:** Stable Diffusion, Dall-E 2 i Midjourney *Imatge extreta de https://www.linkedin.com/pulse/dall-e-midjourney-stable-diffusion-deep-fake-aimldl-image-abdoullaev/*

Mentre la IA encara avança ràpidament, la seva creixent capacitat i autonomia han obert importants debats sobre qüestions ètiques i de control. La irrupció de sistemes com ChatGPT ha portat a experts i empreses tecnològiques a demanar una moratòria per establir estàndards ètics abans de continuar desenvolupant IA més avançada. El futur d'aquesta tecnologia està ple d'oportunitats, però també de riscos que requeriran un acurat equilibri.

## 3.2. CORRENTS PRINCIPALS

La intel·ligència artificial és un vast camp interdisciplinari que ha derivat en nombrosos corrents i enfocaments al llarg de la seva història, cada un aportant tècniques computacionals per replicar diferents aspectes de la intel·ligència. Els principals paradigmes desenvolupats són:

- Intel·ligència Artificial Simbòlica:
  Aquest corrent es basa en l'ús de representacions abstractes del món per raonar sobre el coneixement. Utilitza regles i lògica per manipular símbols que representen idees sobre el món.
  Els sistemes simbòlics intenten imitar el raonament humà mitjançant la manipulació de representacions formals. Requereix que els experts codifiquin manualment grans quantitats de coneixement sobre el domini en formats com regles, ontologies, marcs semàntics, etc.
  Algunes de les tècniques més rellevants d'aquest corrent són: sistemes experts, raonament basat en casos, lògica difusa, raonament automàtic i representació del coneixement.

- Xarxes Neuronals Artificials:
  Inspirades en les interconnexions entre les neurones biològiques, les xarxes neuronals artificials busquen replicar el funcionament massivament paral·lel i distribuït del cervell humà.
  Estan formades per capes de neurones artificials simples però interconnectades entre si, que s'activen mitjançant funcions matemàtiques. Poden aprendre a partir d'exemples sense necessitat de programació explícita. Desenvolupades inicialment en la dècada de 1940 per Frank Rosenblatt. McCulloch i Pitts van desenvolupar models matemàtics de neurones, van ressorgir amb força als anys vuitanta i actualment són el fonament de l'aprenentatge profund, permetent assolir fites en visió artificial, processament de llenguatge natural, conducció autònoma, diagnòstic mèdic i altres àrees.

- Algorismes Genètics:

  Basats en la teoria de l'evolució biològica, els algorismes genètics resolen problemes generant múltiples solucions candidates mitjançant operadors evolutius abstractes com la mutació, l'encreuament i la selecció.

  Són especialment útils per optimitzar i trobar solucions aproximades a problemes massa complexos per a ser resolts analíticament. Tenen aplicacions en disseny industrial, classificació de dades, reconeixement de patrons, predicció econòmica i financera, i molts altres camps.

- Lògica Difusa:

  Desenvolupada per Lotfi Zadeh als anys seixanta, la lògica difusa o borrosa permet tractar amb incertesa, vaguetat i valors entre cert i fals, tal com fa el raonament humà.

  Es fa servir àmpliament en sistemes de control, diagnòstic mèdic, anàlisi d'imatges, modelat geològic, intel·ligència de negocis, sistemes d'informació geogràfica i altres àrees on la incertesa i imprecisió són importants.

- Computació Evolutiva:

  Agrupa un conjunt de tècniques inspirades en la biologia evolutiva, com els algorismes genètics i evolutius, la programació genètica, les estratègies evolutives, la programació evolutiva i la vida artificial.

  Es fan servir per resoldre problemes complexos sense una solució analítica factible, optimitzant paràmetres mitjançant iteracions i conservant les solucions que millor s'ajusten als objectius.

- Agents Intel·ligents:

  Els agents intel·ligents són programes informàtics que perceben el seu entorn mitjançant sensors i actuen de manera autònoma en aquest entorn per assolir objectius.

  Poden exhibir comportaments d'aprenentatge amb l'experiència i fins i tot interaccionar amb humans mitjançant el llenguatge. Es fan servir àmpliament en els videojocs, simulacions, robots, conducció autònoma, comerç electrònic i altres àrees.

- Aprenentatge Automàtic:

  Disciplina centrada a desenvolupar les tècniques computacionals que permetin que les màquines aprenguin i millorin amb l'experiència sense programació explícita.

  Inclou tècniques supervisades i no supervisades, aprenentatge per reforç, profund, probabilístic, basat en instàncies, entre moltes altres. És indispensable en sistemes de recomanació, traducció automàtica, diagnòstic predictiu, conducció autònoma i un amplíssim ventall d'aplicacions.

- Visió per Computador:

  Aquest corrent busca dotar les màquines de la capacitat d'interpretar, processar i comprendre imatges i vídeos del món real de la mateixa manera que ho fa el cervell humà.

  És indispensable per a aplicacions com vehicles autònoms, robots mòbils, inspecció industrial, endoscòpies computaritzades, identificació biomètrica, restauració d'imatges antigues, organització massiva de fotografies, entre moltes altres.

- Processament del Llenguatge Natural:

  La branca del PLN persegueix que les màquines puguin comprendre, interpretar i generar textualment el llenguatge humà. Els sistemes conversacionals interactuen amb persones mitjançant el diàleg.

  Té aplicacions en traducció automàtica, resum i generació de textos, classificació i clustering de documents, analítiques de sentiments, assistents virtuals, centres de trucades, etc.

- Intel·ligència Artificial General:

  Aquest corrent persegueix desenvolupar sistemes integrats d'IA amb el nivell de versatilitat i generalitat característic de la intel·ligència humana.

  Investiga com replicar les capacitats humanes d'abstracció, generalització, raonament transversal i transferència de coneixement entre dominis diferents.

  És la forma d'IA més completa i complexa, i encara dista de ser assolida.

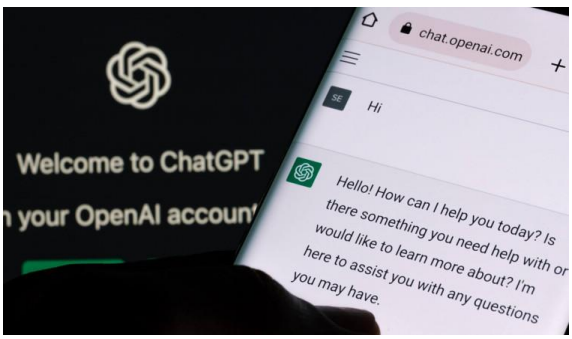
### 3.3. ESTAT ACTUAL

En les últimes dècades els avenços en intel·ligència artificial han estat senzillament espectaculars gràcies a factors com l'enorme augment de la capacitat de computació disponible al consumidor, la possibilitat d'emmagatzemar i processar volums massius de dades en el núvol, i l'aparició de poderosos algoritmes d'aprenentatge profund amb arquitectures de xarxes neuronals cada vegada més complexes. Actualment, la intel·ligència artificial està present en un amplíssim ventall d'aplicacions que impacten pràcticament tots els camps de l'activitat humana.

Alguns dels exemples més destacats inclouen assistents virtuals conversacionals com Siri d'Apple, Alexa d'Amazon o l'Assistent de Google. Són capaços de mantenir diàlegs sorprenentment naturals i fluids amb els usuaris, respondre preguntes sobre pràcticament qualsevol tema d'interès general, realitzar cerques a internet o dispositius connectats i dur a terme diverses accions a petició de l'usuari.

**Imatge 7.** Alexa, Siri y Google Assistant
*Imatge extreta de https://www.youtube.com/watch&v=NhFfWdDvmkY*

Un cas particularment impactant el constitueix ChatGPT, un xatbot desenvolupat per OpenAI capaç de sostenir converses extremadament humanes sobre qualsevol tema imaginable, a més de generar textos de manera coherent sobre un tema donat, escriure poemes o fins i tot programar algoritmes i contingut educatiu a partir d'una descripció en llenguatge natural




**Imatge 8:** ChatGPT *Imatge extreta de https://www.welivesecurity.com/la-es/2023/02/01/formas-utilizar-chatgpt-fines-maliciosos-cibercriminales/*

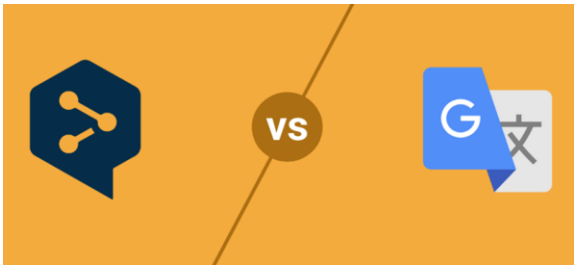
Pel que fa a aplicacions mèdiques de la IA, s'estan desenvolupant sistemes de diagnòstic assistit basats en aprenentatge profund que analitzen dades d'imatge mèdica com radiografies o mamografies, sent capaços de detectar patologies com càncer de pulmó, de mama o fractures òssies amb més precisió i menys taxa d'error que els millors radiòlegs humans.

**Imatge 9:** La intel·ligència artificial en medicina *Imatge extreta de https://topazium.com/covid-19-la-inteligencia-artificial-al-servicio-de-la-medicina/*

En l'àmbit de la conducció autònoma, companyies com Tesla o Waymo han desenvolupat cotxes que poden circular de manera totalment autònoma per entorns urbans i en circumstàncies meteorològiques adverses gràcies als avenços en visió per computador i la capacitat de prendre decisions complexes en temps real.

**Imatge 10:** Autopilot de Tesla. Imatge extreta de: https://www.autobild.es/noticias/video-tesla-autopilot-cruza-paso-peatones-toda-velocidad-pesar-sistema-detecta-persona-1247090

Pel que fa a la traducció automàtica, serveis en línia com Google Translate o DeepL permeten mantenir converses en temps real traduint entre centenars d'idiomes diferents amb una qualitat extremadament propera a l'humà.




**Imatge 11:** DeepL vs Google Translate: Which Is Better?. Imatge extreta de: https://translatepress.com/deepl-vs-google-translate-comparison/

Els gegants digitals com Amazon o Netflix utilitzen sofisticats sistemes de recomanació personalitzats entrenats sobre dades de consum i preferències de milions d'usuaris.


**Imatge 12:** Simplificació dels sistemes de recomanació personalitzats. Imatge extreta de: https://www.statdeveloper.com/introduccion-a-los-sistemas-de-recomendacion/

I en l'àmbit de la robòtica humanoide, companyies com Agility Robotics, Boston Dynamics o Digit de Ford han desenvolupat robots bípedes amb increïble precisió motriu, equilibri, destreses manipulatives i fins i tot certa capacitat per navegar entorns dinàmics sense col·lidir amb obstacles.

**Imatge 13:** Spot, uns dels millors robots de BostonDynamics. Imatge extreta de: https://www.iotworldtoday.com/robotics/boston-dynamics-spot-the-design-behind-the-robot-dog

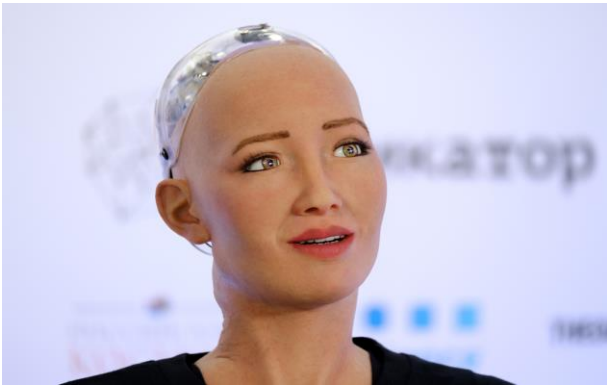
Les aplicacions pràctiques d'IA avui dia ja estan transformant profundament nombrosos aspectes de la nostra manera de viure tal com: comunicar-nos, consumir, produir, desplaçar-nos o gaudir del temps lliure. I és esperable que aquest impacte continuï amplificant-se de forma exponencial durant les pròximes dècades.

### 3.4. FUTUR I REPTES

La intel·ligència artificial es troba encara en una fase primerenca de desenvolupament, amb enormes avenços per recórrer abans d'assolir les capacitats generals i completes de la intel·ligència humana. Els experts preveuen progressos accelerats en les pròximes dècades encara que amb incertesa sobre quan es podrien aconseguir fites clau.

A curt termini s'espera que la IA continuï millorant en àrees on ja ha demostrat gran progrés, com el processament del llenguatge, la visió artificial, el reconeixement de veu i la robòtica. S'integrarà més en aplicacions quotidianes, automòbils, dispositius electrònics, sistemes de seguretat i altres camps.

A mitjà termini, podrien aparèixer les primeres màquines amb intel·ligència i consciència comparables a les d'un ésser humà, encara que limitades a dominis o tasques específiques. Aquesta fita és coneguda com a IA forta o artificial general. Implicaria reptes ètics sense precedents-



**Imatge 13:** Sophia, el primer robot con ciudadanía. Imatge extreta de: https://criptotendencia.com/2018/08/17/sophia

A llarg termini, la meta més ambiciosa és obtenir una IA artificial general il·limitada, equivalent a la intel·ligència humana. Això plantejaria preguntes existencials sobre el lloc dels humans en una civilització amb màquines superintel·ligents. Alguns experts creuen possible una singularitat tecnològica en què la IA es millorés a si mateixa de forma explosiva.

Per assolir aquest potencial la intel·ligència artificial haurà de superar reptes fonamentals:

- Comprendre veritablement el llenguatge, no només processar-lo estadísticament. Interpretar tots els matisos i el context.
- Adquirir sentit comú, és a dir, tot el coneixement implícit que els humans adquirim sobre com funciona el món.
- Transferir aprenentatges entre dominis diferents, no només dominar una especialitat. Raonar de manera general i transversal.
- Desenvolupar una veritable comprensió visual del món en tres dimensions, no només classificar imatges.
- Aconseguir una autèntica intel·ligència artificial creativa, capaç de generar art, música i idees originals.
- Desenvolupar intel·ligència emocional, per interpretar i expressar emocions com els humans.
- Garantir que els sistemes d'IA siguin ètics, transparent i responsables, alineant-los amb els valors humans.
- Resoldre els reptes computacionals d'emmagatzematge, potència de càlcul i eficiència energètica necessaris per executar models d'IA en entorns reals.

El futur de la IA conté immenses promeses per millorar la vida humana però també riscos existencials. Cal investigació prudent i un marc ètic i de control democràtic sobre l'ús d'aquesta tecnologia innovadora. És un dels majors desafiaments de la nostra civilització en les dècades vinents.

# 4. ELS AGENTS INTEL·LIGENTS

Els agents intel·ligents són programes que perceben el seu entorn i actuen de manera autònoma en ell per assolir uns objectius. Constitueixen un paradigma important dins la intel·ligència artificial, amb aplicacions en àrees com els videojocs, la robòtica, la conducció autònoma i els assistents virtuals.

Els agents intel·ligents es caracteritzen per la seva autonomia, capacitat de percebre l'entorn, d'actuar en ell, d'orientar-se a metes i d'adaptar el seu comportament amb l'experiència. Poden exhibir diferents nivells d'autonomia, capacitats d'aprenentatge, des d'agents purament reactius fins a agents complets que persegueixen objectius abstractes a llarg termini.

En aquest apartat s'abordaran amb més detall la definició i principals característiques dels agents intel·ligents, els seus models, i les seves múltiples aplicacions pràctiques. L'objectiu és comprendre la rellevància d'aquest paradigma dins el camp de la intel·ligència artificial.

## 4.1. DEFINICIÓ I CARACTERÍSTIQUES

Els agents intel·ligents són programes informàtics dissenyats per exhibir un comportament flexible i adaptatiu que els permeti assolir objectius en entorns complexos i canviants. Es defineixen formalment com a sistemes situats en un entorn concret, amb capacitat de percebre aquest entorn a través de sensors i actuar de forma autònoma sobre ell per aconseguir fites específiques.

Els agents intel·ligents es diferencien dels programes tradicionals, que segueixen un conjunt predeterminat d'instruccions, en el fet que són capaços de prendre decisions pròpies en funció de l'experiència per adaptar-se a entorns dinàmics. Poden modificar la seva estratègia i comportament sobre la marxa per maximitzar l'assoliment dels seus objectius.

Una característica definitòria dels agents intel·ligents és la seva autonomia. Un cop dissenyats i posats en marxa, funcionen sense intervenció humana directa. Se'ls defineixen objectius inicials i paràmetres bàsics de funcionament, però a partir d'aquí actuen de forma independent en funció de l'estat percebut de l'entorn. Prenen decisions per pròpia iniciativa sense necessitar supervisió externa.

Per poder actuar de forma autònoma, els agents intel·ligents han de ser capaços de percebre i representar el seu entorn. Per això incorporen sensors que capturen dades del món real o digital que els envolta de forma contínua. Poden ser sensors físics en agents robòtics o sensors de dades en agents software. La informació percebuda els permet mantenir un model actualitzat de l'estat de l'entorn.

A més de percebre l'entorn, els agents intel·ligents poden actuar sobre ell provocant canvis en el seu estat. Per exemple, un agent robot pot desplaçar objectes, prémer interruptors o dur a terme tasques, mentre un agent software pot afegir, modificar o esborrar dades d'una base de dades o un sistema. Les accions executades pels agents alteren l'entorn d'alguna manera.

Els agents seleccionen les seves accions buscant maximitzar l'assoliment dels objectius que se'ls hagi definit prèviament. Poden gestionar i prioritzar múltiples objectius alhora, escollint en cada moment les accions que els semblin més adequades per avançar cap a les metes marcades. La persecució d'objectius és el que guia el seu comportament.

Addicionalment, els agents intel·ligents poden respondre de forma flexible i en temps real als esdeveniments que ocorrin a l'entorn. Sense necessitat d'esperar instruccions externes, reaccionen immediatament als canvis percebuts per tractar de mantenir el progrés cap als objectius. Aquesta capacitat de resposta els permet actuar de forma reactiva i prendre decisions en temps real.

Una altra característica important és l'adaptabilitat. Els agents no segueixen un comportament completament predeterminat, sinó que poden modificar les seves estratègies sobre la base de l'experiència prèvia per incrementar la seva efectivitat. Identifiquen situacions ja viscudes i seleccionen les accions que van funcionar millor.

Això els permet optimitzar els mètodes per assolir els objectius segons les particularitats de l'entorn.

A més, prenen en consideració el context específic de cada moment per escollir la millor resposta possible davant de cada situació particular. Per exemple, un agent conversacional s'adaptarà al to emocional i vocabulari de cada usuari. La capacitat d'ajustar-se a les circumstàncies concretes és un tret definitori dels agents intel·ligents.

Els agents poden comunicar-se i interactuar amb altres agents o humans per intercanviar informació utilitzant diferents tipus de llenguatges, des de protocols formals fins a llenguatge natural. La comunicació els permet coordinar-se i col·laborar per assolir objectius complexos.

### 4.2. MODELS I ARQUITECTURES

Els agents intel·ligents es poden implementar computacionalment mitjançant diferents models, que determinen les seves capacitats cognitives, i arquitectures, que defineixen la seva estructura interna.

Els agents reactius són els més senzills, i seleccionen accions mitjançant regles condicionals que relacionen percepcions amb accions, sense tenir en compte experiències passades ni disposar de representacions explícites del món. Aquest model és apropiat per a entorns molt simples i predictibles, i un exemple el constitueixen els agents situats en algunes simulacions de trànsit.

Els agents amb memòria limitada incorporen informació sobre percepcions i accions passades, la qual cosa els confereix major context per decidir, encara que la memòria sigui de curt abast. Un exemple són els agents en jocs amb mons parcialment observables.

Els agents basats en objectius, a més de reaccionar al seu entorn, incorporen objectius explícits definits prèviament que guien el seu comportament i els permeten prendre decisions buscant maximitzar l'assoliment d'aquests objectius. Aquests

objectius a llarg termini es descomponen en subobjectius a curt termini més manejables. Un exemple són els agents en videojocs que intenten augmentar la seva puntuació.

Els agents basats en utilitat assignen valors numèrics anomenats utilitats a les diferents accions, percepcions i objectius possibles, i seleccionen accions buscant maximitzar la utilitat total esperada. Això els permet incorporar incertesa mitjançant l'ús de probabilitats. Un exemple són els agents d'inversió financers.

Finalment, els agents complets són els més sofisticats i poden representar coneixement complex sobre el món, raonant lògicament per establir plans estratègics d'alt nivell i polítiques generals per assolir objectius abstractes.

Pel que fa a les arquitectures, aquestes poden ser basades en regles, on l'agent pren decisions mitjançant la combinació de regles establertes manualment; basades en xarxes neuronals profundes, que permeten determinar les accions sense dissenyar regles explícites; o híbrides, combinant regles amb aprenentatge automàtic. L'elecció de l'arquitectura dependrà de l'entorn, la complexitat del problema i els recursos disponibles.

### 4.3. TÈCNIQUES D'APRENENTATGE

Els agents intel·ligents requereixen mecanismes d'aprenentatge per millorar les seves habilitats de percebre, raonar i actuar de forma autònoma en entorns complexos. Hi ha diverses tècniques d'aprenentatge automàtic que s'han aplicat amb èxit en agents intel·ligents:

- Aprenentatge per reforç: L'agent aprèn interaccionant amb l'entorn i rebent senyals de reforç (recompenses o penalitzacions) que guien l'optimització del seu comportament. Permet aprendre sense exemples etiquetats. S'ha combinat amb èxit amb xarxes neuronals profundes per crear agents que aprenen habilitats directament des de dades sensorials.

- Aprenentatge per imitació: L'agent observa i imita el comportament d'un expert, normalment humans. Permet adquirir noves habilitats de forma molt eficient. S'ha aplicat en robots que aprenen habilitats motrius a partir de demostracions.

- Aprenentatge supervisat: L'agent entrena models predictius a partir d'exemples etiquetats d'entrades i sortides desitjades. Els models entrenats permeten generalitzar a noves dades. S'usa en components d'agents per a tasques específiques (visió, reconeixement de veu, predicció).

- Aprenentatge no supervisat: L'agent identifica patrons i relacions en grans volums de dades. Permet detectar agrupaments, reduir dimensionalitat i trobar associacions. Útil per a preprocessament de dades sensorials.

- Aprenentatge per transferència: Reutilitza coneixement d'una tasca en una de nova. Permet entrenar agents més ràpids en nous entorns relacionats amb experiències passades.

Els mecanismes d'aprenentatge són una peça clau en l'arquitectura dels agents intel·ligents, ja que permeten millorar les seves habilitats perceptives, de raonament, planificació i presa de decisions de forma contínua a mesura que interaccionen amb l'entorn.

L'aprenentatge per reforç i l'aprenentatge per imitació permeten entrenar de forma seqüencial i sense supervisió agents cada vegada més capacitats. L'aprenentatge supervisat pot millorar components concrets. La transferència accelera l'aprenentatge en nous entorns.

Les tècniques d'aprenentatge automàtic són essencials per dotar als agents intel·ligents la capacitat d'adquirir, millorar i transferir habilitats de forma autònoma al llarg del temps. Aquesta propietat és clau per assolir agents artificials realment intel·ligents.

# 5. DISSENY I IMPLEMENTACIÓ D'UN AGENT INTEL·LIGENT

Minecraft és possiblement el videojoc sandbox 3D més popular de la història. Permet als jugadors explorar mons pràcticament infinits, recollir recursos i construir tota mena d'estructures i mecanismes. Els jugadors comencen el joc amb res i han de sobreviure collint fusta i menjar, fabricant eines bàsiques, construint refugis i eventualment progressant a través d'un arbre tecnològic que els permet crear eines, armes i armadures cada cop més avançades.

**Imatge 13:** Minecraft. *Imatge pròpia*

Aquest entorn obert representa tot un repte per a la IA. Els agents necessiten ser capaços de navegar per entorns 3D complexos, reconèixer i recollir recursos rellevants, planificar i dur a terme accions seqüencials com cultivar, cuinar, fabricar objectes o construir estructures i, en general, prendre decisions informades en un entorn dinàmic.

En aquest projecte es presenta CraftyAI, un agent dissenyat per jugar a Minecraft de forma completament autònoma. L'objectiu principal és que CraftyAI sigui capaç d'explorar el món de manera sistemàtica, descobrir i recollir recursos, fabricar eines i objectes, construir estructures, lluitar contra monstres, cultivar i, en general, sobreviure i prosperar en el joc indefinidament com ho faria un jugador humà.

Per assolir aquest objectiu, CraftyAI està format per tres components principals:

- Un planificador automàtic de tasques que proposa objectius adequats al nivell actual de l'agent tenint en compte el seu progrés d'exploració. Aquest planificador està basat en el model de llenguatge GPT-4.

- Una biblioteca d'habilitats on es guarden arxius JSON que representen comportaments complexos apresos prèviament. Aquests arxius poden ser reutilitzats i combinats per resoldre noves tasques.

- Un mecanisme iteratiu de prompting que permet refinar i verificar els programes generats per GPT-4 a través de feedback de l'entorn i errors d'execució.

CraftyAI interactua amb l'entorn de Minecraft a través de l'API Mineflayer, que permet controlar un bot de manera programàtica. Tot el codi està implementat en Python i s'executa sobre l'entorn de desenvolupament Visual Studio Code.


**Imatge 14:** CraftyAI en Minecraft. *Imatge pròpia*

Aquest enfocament basat en codi executable i models de llenguatge permet que l'agent exhibeixi un comportament més sistemàtic, interpretable i composicional en comparació amb mètodes basats en accions de baix nivell. A més, el fet de construir progressivament una biblioteca reutilitzable d'habilitats alleuja el problema de l'oblit catastròfic en aprenentatge continuat.

També presenta nombrosos avantatges respecte a l'aprenentatge per reforç tradicional amb accions primitives:

- Permet representar accions complexes i seqüencials, essencials en Minecraft.
- Els programes són interpretables i composicionals, facilitant l'aprenentatge acumulatiu.
- S'evita l'oblit catastròfic típic en entorns continus.
- No requereix entrenament explícit amb gradients o ajust de paràmetres.

En aquest apartat es detallen els objectius de disseny, l'arquitectura, els algoritmes, les tècniques d'IA i altres aspectes d'implementació de CraftyAI. Els resultats experimentals que demostren les capacitats d'aquest agent es presenten en l'apartat següent.

### 5.1. OBJECTIUS I REQUISITS

L'objectiu principal de CraftyAI és ser capaç de jugar a Minecraft de forma completament autònoma i amb un comportament similar al d'un jugador humà. Per tant, els requisits funcionals principals són:

- Exploració autònoma: l'agent ha de ser capaç de navegar pel món de Minecraft, evitant perills, i explorant de forma sistemàtica per trobar nous biomes, estructures i recursos.

- Recol·lecció de recursos: identificar, prioritzar i recollir eficientment els recursos necessaris en cada moment, com fusta, menjar, mineral, etc.

- Fabricació d'objectes: usar els recursos recollits per fabricar les eines, armes, armadures i objectes necessaris per progressar en el joc. Ha de ser capaç d'utilitzar taules d'enceballs, forns i altres blocs de fabricació.

- Construcció: construir estructures bàsiques com refugis, granges o portals al Nether. Les construccions han de ser funcionals i útils per protegir o emmagatzemar recursos.

- Supervivència: mantenir uns nivells adequats de vida i gana, evitant o combatent monstres si és necessari. Prioritzar accions que millorin la supervivència.

- Mineria: trobar i extreure eficientment minerals valuosos com ferro, diamants o redstone que permetin elaborar objectes avançats. Ha de minar en coves i a diferents nivells de profunditat.

- Agricultura: ser capaç de cultivar diversos tipus de conreus, recollir-los, replantar i fer servir els cultius.

- Criança: criar i recollir recursos d'animals com vaques, ovelles o pollastres. Construir granges adequades.

- Combat: lluitar eficaçment contra monstres com aranyes, zombis o esquelets fent servir les millors armes i armadures disponibles.

- Gestió d'inventari: gestionar l'espai limitat de l'inventari de forma intel·ligent, descartant objectes innecessaris i prioritzant els més valuosos.

- Ús eficient de recursos: optimitzar l'ús de recursos escassos com menjar, torxes o eines amb durabilitat limitada.

- Progrés eficient: avançar en el joc superant reptes i aconseguint fites de forma eficient, sense repetir tasques innecessàriament.

Per assolir aquests requisits funcionals, CraftyAI necessita una sèrie de capacitats:

- Percepció en 3D de l'entorn immediat per identificar recursos, perills i punts d'interès.

- Planificació a llarg termini per establir objectius i prioritats.

- Raonament seqüencial per descompondre objectius complexos en subtasques.

- Presa de decisions contextual en temps real tenint en compte l'entorn i l'estat intern.

- Aprenentatge per reforç per millorar les polítiques de control.

- Memòria per recordar localitzacions i patrons d'interacció amb l'entorn al llarg del temps.

- Representacions vectorials d'altes dimensions per codificar coneixement sobre el món.

- Mecanismes d'atenció visual per explorar escenes 3D de forma eficient.

L'arquitectura de CraftyAI està dissenyada específicament per proporcionar aquestes capacitats d'una forma integral utilitzant els últims avenços en aprenentatge profund i models de llenguatge.

## 5.2. ARQUITECTURA

CraftyAI seguirà una arquitectura modular per separar les diferents capacitats en components especialitzats. Això permetrà un disseny flexible i escalable. L'arquitectura contindrà els següents mòduls principals:

- **Mòdul de percepció**: s'encarregarà de rebre les observacions de l'entorn Minecraft a través de l'API Mineflayer i processar-les per generar una representació estructurada de l'estat del món. Aquesta representació inclourà elements com l'inventari, blocs i entitats properes, posició del jugador, bioma actual, etc.

- **Mòdul de raonament**: contindrà el model de llenguatge GPT-4. Rebent la representació de l'entorn generada pel mòdul de percepció, raonarà sobre l'estat actual i generarà plans d'alt nivell per assolir objectius utilitzant codi executable com a llenguatge d'acció.

- **Mòdul de control**: s'encarregarà d'interpretar el codi generat pel mòdul de raonament i traduir-lo a les corresponents crides a l'API de Mineflayer per executar accions motores de baix nivell.

- **Mòdul d'aprenentatge**: incorporarà l'algoritme de Q-learning basat en xarxes neuronals convolucionals per millorar la política d'acció de l'agent de forma no supervisada a través de l'experiència.

- **Mòdul de memòria**: emmagatzemarà les habilitats apreses en forma de programes executables, permetent la seva reutilització i composició. La memòria persistirà entre episodis d'aprenentatge.

- **Mòdul de currículum**: generarà tasques i fites adequades per l'agent en cada moment, permeten una exploració sistemàtica i progressiva de l'entorn.

Els mòduls estaran connectats a través d'una arquitectura pipeline on la sortida d'un mòdul alimenta l'entrada del següent. Per exemple, el mòdul de percepció proporciona representacions d'estat al planificador, el qual genera plans executats pel controlador.

A més, hi haurà connexions recurrents, on els mòduls posteriors proporcionen retroalimentació als anteriors. Per exemple, els errors d'execució del controlador s'enviaran al mòdul de raonament per millorar els futurs plans.

Aquesta arquitectura modular dotarà CraftyAI de les capacitats de percebre, raonar, actuar i aprendre de forma integrada per mostrar un comportament cada vegada més competent en Minecraft de manera completament autònoma.

### 5.3. DETALLS D'IMPLEMENTACIÓ

Per a la implementació de CraftyAI s'utilitzarà Python com a llenguatge de programació principal, per la seva flexibilitat, capacitat per a càlcul numèric i disponibilitat de llibreries d'aprenentatge profund.

L'entorn de simulació serà Minecraft, accedint-hi a través de l'API Mineflayer. Aquesta API permetrà que l'agent percebi l'estat del joc en temps real i executi accions motores de baix nivell. En concret, es faran servir funcions com bot.lookAt(), bot.placeBlock(), bot.collectBlock(), bot.fish(), etc. per interactuar amb l'entorn.

El mòdul de percepció processarà la informació sensorial rebuda de Mineflayer per crear una representació estructurada de l'entorn, incloent-hi elements com l'inventari, blocs i entitats properes, bioma actual, nivell de vida, etc. Aquesta representació servirà com a entrada per al mòdul de planificació.

El mòdul de planificació contindrà el model de llenguatge GPT-4, al qual s'alimentarà la representació de l'entorn per obtenir plans d'acció d'alt nivell en forma de codi executable. Aquest codi serà transmès al mòdul d'execució.

El mòdul d'execució interpretarà el codi rebut del planificador i el traduirà a les corresponents crides a l'API de Mineflayer per dur a terme les accions. També capturarà els errors d'execució i el feedback de l'entorn generat durant l'execució.

El model d'aprenentatge serà una xarxa neuronal convolucional per a l'estimació de Q-valors, entrenada mitjançant l'aprenentatge de reforç profund. La xarxa prendrà com a entrada la representació de l'entorn i retornarà valors Q per a cada acció possible. Aquests valors Q permetran que el mòdul de planificació seleccioni accions òptimes.

La xarxa neuronal s'entrenarà de forma supervisada en un entorn simulat de Minecraft, utilitzant l'exploració aleatòria i l'algoritme de Q-Learning per millorar la política. L'objectiu de recompensa serà maximitzar elements nous descoberts i distància recorreguda.

Addicionalment, es faran servir tècniques com experiència replay i target networks per estabilitzar i millorar l'aprenentatge. La xarxa entrenada es congelarà i s'integrarà en l'agent perquè guiï l'exploració en entorns Minecraft desconeguts.

CraftyAI integrarà percepció, planificació amb models de llenguatge, execució de codi i aprenentatge profund per mostrar un comportament cada vegada més competent en l'entorn dinàmic i obert de Minecraft.

# 6. ENTORN DE SIMULACIÓ DE L'AGENT INTEL·LIGENT EN MINECRAFT

Perquè un agent intel·ligent com CraftyAI pugui aprendre i desenvolupar les seves habilitats de forma completament autònoma, és fonamental comptar amb un entorn de simulació adequat. L'entorn ideal ha de presentar una gran varietat de situacions, reptes i tasques amb diferents nivells de complexitat perquè l'agent pugui explorar, experimentar i millorar contínuament.

Alhora, l'entorn ha de proporcionar retroalimentació informativa sobre les conseqüències de les accions de l'agent, permetent associar decisions amb resultats i reforçar els comportaments desitjats. Finalment, l'entorn ha d'oferir mecanismes d'interacció flexibles perquè l'agent pugui percebre l'estat actual i executar accions motores de manera programàtica.

El videojoc Minecraft representa un entorn virtual excel·lent que compleix amb aquests requisits. Minecraft és un món sandbox obert amb possibilitats pràcticament il·limitades per construir, explorar, recol·lectar recursos i interactuar amb altres entitats. Els jugadors poden desbloquejar arbres tecnològics que requereixen habilitats cada cop més sofisticades, la qual cosa impulsa l'aprenentatge i el progrés.

A més, Minecraft proporciona retroalimentació natural sobre l'estat del món i l'inventari del jugador després de cada acció. Per exemple, destruir un bloc de fusta proporciona fusta, mentre que minar pedra genera pedra tosca. Aquesta predictibilitat permet a l'agent associar decisions amb conseqüències de manera fiable.

Finalment, Minecraft ofereix diverses API i llibreries que habiliten la connexió d'agents externs. En aquest treball s'utilitzarà específicament l'API Mineflayer, que permetrà a CraftyAI percebre l'estat del joc en temps real i executar accions motores de baix nivell de manera programàtica.

L'entorn virtual, dinàmic i obert de Minecraft, la seva retroalimentació informativa i les seves interfícies d'interacció el converteixen en la plataforma ideal per al desenvolupament i l'avaluació de CraftyAI com un agent autònom capaç d'aprendre contínuament.

## 6.1. DESCRIPCIÓ DE L'ENTORN DE MINECRAFT

Minecraft és un joc de sandox obert en 3D on els jugadors exploren mons generats procedimentalment i interactuen amb el seu entorn. Presenta les següents característiques principals:

- Mapes oberts i virtualment infinits. Els mons de Minecraft són 3D i es generen procedimentalment a mesura que el jugador explora, la qual cosa permet una àrea de joc potencialment il·limitada.

- Blocs i estructures. Els mapes estan formats per blocs individuals apilats en una graella 3D, que representen diferents materials com terra, pedra, minerals, aigua, lava, etc. Els jugadors poden col·locar i destruir blocs.

- Biomes variats. Hi ha diferents biomes o ambients, com deserts, jungles i tundres nevades. Cadascun té els seus propis blocs i recursos.

- Cicle dia/nit. Completar un cicle dia/nit dura 20 minuts reals. De nit apareixen criatures hostils.

- Física bàsica. Existeixen la gravetat, inèrcia i altres efectes físics que afecten el moviment del jugador i objectes.

- Recursos. Es poden obtenir i recollir recursos com fusta, pedra, minerals, etc. Molts es troben en biomes específics.

- Sistema d'artesania. Els recursos es poden combinar en una taula d'artesania per crear eines, armes, armadures i centenars d'altres ítems seguint diverses receptes.

- Arbre tecnològic. Hi ha diferents nivells d'eines i equipaments (fusta, pedra, ferro, diamant) que requereixen materials i artesania progressivament més complexes.

- Entitats interactives. Apareixen animals pacífics i criatures hostils. Es pot comerciar amb vilatans o lluitar contra monstres.

Aquestes característiques principals, especialment el seu caràcter obert, dinàmic i complex, fan que Minecraft sigui l'entorn ideal per al desenvolupament d'agents intel·ligents autònoms.

### 6.2. EINES I LLIBRERIES

Per poder desenvolupar un agent com CraftyAI que interactuï de forma autònoma en l'entorn virtual de Minecraft, és fonamental comptar amb les interfícies de programació i llibreries adequades que proporcionin abstracció i control sobre el joc.

Idealment, aquestes eines han de permetre als desenvolupadors connectar amb el client de Minecraft, obtenir informació en temps real sobre l'estat del joc, i enviar accions i comandaments per controlar un personatge i interactuar amb el món de manera flexible i dinàmica.

Després d'avaluar diferents opcions, s'ha seleccionat l'API Mineflayer per desenvolupar CraftyAI, ja que actualment és una de les interfícies més completes i potents per crear bots de Minecraft.

Mineflayer és una llibreria de codi obert implementada en JavaScript, la qual cosa la fa molt àgil per prototipar i experimentar. Tanmateix, existeixen enllaços oficials per utilitzar-la també des de Python, el que serà ideal per al backend d'aprenentatge de màquina de CraftyAI.

A continuació es detallen algunes de les capacitats més rellevants que Mineflayer posa a disposició per desenvolupar agents:

- Permet connectar amb un client Minecraft utilitzant un client bot i unir-se a mons en execució de manera programàtica.

- Dona accés a tota la informació d'estat del joc en temps real, incloent-hi posició del jugador, inventari complet, estadístiques de vida, entitats i blocs propers, receptes d'artesania disponibles, contingut de cofres, etc.

- Permet el control complet del personatge, amb funcions per moure's, girar, saltar, obrir inventari, seleccionar objectes, atacar entitats, usar objectes, entre moltes altres accions motores de baix nivell.

- Proporciona esdeveniments i crides de retorn per reaccionar a tota mena d'interaccions i esdeveniments en el joc, com recollir objectes, rebre dany, matar mobs, obrir cofres, ploure, fer-se de dia, etc.

- Inclou pathfinding i seguiment d'entitats integrades per moure's de forma intel·ligent cap a destinacions o seguir altres criatures.

- Permet minar de forma eficient blocs específics i recollir els objectes que contenen de forma automàtica.

- Conté helpers per gestionar fàcilment l'inventari, obrir/tancar cofres i transferir objectes.

- Disposa d'una API per articular tasques d'artesania complexes a partir d'ingredients i receptes.

Mineflayer proporciona tot el necessari per percebre, raonar i actuar en Minecraft de forma fiable i eficient. La seva API de baix nivell permetrà dotar CraftyAI de tota la potència, flexibilitat i control necessaris per aprendre habilitats generals que transfereixin al món real.

# 7. EXPERIMENTACIÓ I RESULTATS

Un cop dissenyat i implementat l'agent CraftyAI, el pas següent en aquest treball és avaluar experimentalment el seu comportament i rendiment en l'entorn virtual de Minecraft.

Específicament, els experiments busquen validar que CraftyAI compleix els objectius plantejats i demostra les capacitats desitjades d'exploració autònoma, aprenentatge i millora contínua de les seves habilitats en un entorn dinàmic i complex.

Per a això, es col·locarà CraftyAI en diversos mons de Minecraft generats aleatòriament i se'l deixarà interactuar de forma completament autònoma durant llargs períodes de temps. S'anirà monitorant el seu progrés i les fites assolides en àrees com:

- Exploració de biomes diversos.
- Descobriment de nous objectes i blocs.
- Domini de l'arbre tecnològic de Minecraft.
- Increment en la complexitat i qualitat dels comportaments exhibits.
- Millora sostinguda de les habilitats al llarg del temps.

Addicionalment, es compararan quantitativament els resultats de CraftyAI amb altres agents i tècniques de l'estat de l'art, per validar la seva superioritat en àrees d'interès.

Els resultats mostraran fins a quin punt CraftyAI és capaç d'aprendre de forma completament autònoma en un entorn virtual complex i dinàmic com Minecraft. Les conclusions serviran per analitzar el seu rendiment, les limitacions actuals i el potencial de millora en treballs futurs.

A continuació es detallen els experiments realitzats, els resultats obtinguts i l'anàlisi de les capacitats exhibides per CraftyAI en l'entorn Minecraft.

## 7.1. EXPERIMENTS INICIALS

Per tal d'avaluar les capacitats inicials de CraftyAI i verificar el correcte funcionament dels seus mòduls, es van realitzar una sèrie d'experiments controlats en entorns simplificats de Minecraft.

En una primera fase, es va crear un món en Minecraft on CraftyAI apareixia. En aquest entorn, CraftyAI va ser capaç de percebre correctament elements propers, utilitzar l'inventari, manejar les mecàniques bàsiques del joc i completar tasques individuals mitjançant codis generats per GPT-4 i executats a través de l'API de Mineflayer.

Per exemple, quan se li assignava la tasca de tallar fusta, CraftyAI explorava fins a localitzar un tronc d'arbre proper, s'hi aproximava, s'equipava d'una destral i la feia servir per colpejar el tronc fins a obtenir fusta, mostrant així una execució reeixida de la tasca assignada. Però, resultava que quan acabava de tallar l'arbre, donava error i es parava l'agent. Això va ser solucionat després d'arreglar les API de l'agent.


**Imatge 15:** L'error donat després de tallar l'arbre. *Imatge pròpia*

De forma similar, CraftyAI va demostrar ser capaç de picar pedres, fabricar eines bàsiques, i recollir els objectes resultants, seguint en tots els casos plans generats per GPT-4 a partir de la descripció d'alt nivell de la tasca a realitzar.


**Imatge 16:** CraftyAI tallant arbre. *Imatge pròpia*

Aquestes proves inicials en entorns controlats van permetre validar la correcta integració entre els mòduls de percepció, planificació i execució de CraftyAI, així com el funcionament del bucle d'interacció amb GPT-4, abans de procedir a avaluar-lo en entorns més complexos i dinàmics.

Els experiments van establir un punt de partida sòlid, confirmant que CraftyAI posseeix les capacitats d'interpretar descripcions de tasques en llenguatge natural, generar plans executables, interactuar amb l'entorn Minecraft i completar accions bàsiques, requisits essencials abans d'intentar fites més ambicioses.

### 7.2. ANÀLISI DEL COMPORTAMENT

Després de la prova inicial, el següent pas va ser avaluar el comportament de CraftyAI en món obert de Minecraft generat aleatòriament. L'agent va ser situat en aquell món i se'l va deixar interactuar de forma completament autònoma durant llargs períodes de temps, mentre es prenia nota de les accions realitzades.

En general, es va observar que CraftyAI mostrava uns patrons de comportament coherents i objectius dirigits cap a progressar en el joc. Per exemple, en els seus primers passos solia exercir accions esperables com tallar fusta, fabricar eines bàsiques de fusta i recollir recursos essencials accessibles.

Posteriorment, avança cap a objectius més complexos com minar pedra, fabricar eines de pedra d'eficiència superior, caçar animals per obtenir aliments i explorar en cerca de minerals valuosos com carbó o ferro. Les seves accions demostraven planificació i l'habilitat d'establir metes intermèdies útils per progressar.


**Imatge 17:** CraftyAI picant pedra. *Imatge pròpia*

Addicionalment, es va analitzar que CraftyAI exhibia alguns comportaments interessants, com evitar caure d'altures, nedar cap a la superfície en quedar submergit, o retrocedir quan s'enfronta a un nombre massa gran d'enemics alhora.

L'agent també mostrava una certa curiositat i proactivitat, explorant el mapa en diferents adreces en cerca de nous descobriments. Tanmateix, en algunes ocasions les seves accions també denoten una falta d'objectius a més llarg termini. A vegades, GPT-4 donava el codi de manera incorrecta i CraftyAI es va rendir després de 4 intents.

En alguns casos també es van observar limitacions en les capacitats de planificació a llarg termini de CraftyAI, perquè les seves accions no seguien una progressió constant i es distreia de vegades amb activitats no productives. Establir objectius més globals i mantenir un progrés sostingut continua sent un repte.

Els experiments van evidenciar les competències bàsiques de CraftyAI en àrees com l'exploració, planificació estratègica bàsica, cura de les necessitats vitals i reacció a situacions perilloses. També van revelar àrees de millora en planificació a llarg termini i curiositat sostinguda.

L'anàlisi del comportament de CraftyAI va proporcionar idees valuoses sobre el seu nivell actual d'habilitats i els aspectes a millorar mitjançant un aprenentatge més profund. Els reptes analitzats van servir per guiar el desenvolupament de l'agent en la següent fase.

### 7.3. RESULTATS

Per avaluar completament les capacitats de CraftyAI, es van fer experiments en els quals es van monitorar i analitzar múltiples mètriques d'interès al llarg de centenars d'iteracions en un món obert.

- **Exploració autònoma**

  Una de les principals mètriques va ser el recompte d'elements únics descoberts per CraftyAI al llarg del temps. Després de 200 iteracions, CraftyAI havia trobat un total de 102 elements únics, incloent-hi fusta, pedra i carbó en les primeres iteracions, però progressant ràpidament cap a descobriments més complexos com menjar cuinat, eines de ferro, armes d'or, etc.

  L'anàlisi en detall del progrés iteració a iteració va mostrar que CraftyAI exhibeix una curiositat i proactivitat sostingudes. En gairebé totes les iteracions realitzava accions dirigides a expandir les àrees explorades i obtenir nous recursos, resultant en un creixement constant dels elements descoberts.

- **Progrés eficient**

  Es va monitorar específicament el progrés de CraftyAI en desbloquejar fites importants en l'arbre tecnològic de Minecraft, com la fabricació d'eines de fusta, pedra i ferro. CraftyAI va demostrar una notable eficiència. A més, CraftyAI va recórrer en total més de 4800 blocs de distància. Això li va permetre explorar amb èxit una àmplia varietat de biomes, incloent-hi deserts, jungles i tundres.

- **Generalització de coneixement**

  Per avaluar la capacitat d'abstracció i generalització de coneixements de CraftyAI, se'l va posar a prova en tasques totalment noves no intentades durant el seu entrenament, com elaborar un pic de diamant o trobar una cova.

  En tots els casos, CraftyAI va ser capaç de completar les tasques desconegudes de forma eficient, demostrant que posseeix un model del món que li permet planificar i prendre decisions per assolir objectius arbitraris.

Amb els experiments realitzats i els resultats obtinguts, ens permet extreure una sèrie de conclusions rellevants sobre les capacitats exhibides per CraftyAI:

- CraftyAI demostra ser un agent autònom capaç d'explorar entorns complexos 3D com Minecraft de forma no supervisada. Els seus mecanismes d'exploració li permeten descobrir una gran varietat d'elements sense necessitat de recompenses externes.

- La seva arquitectura basada en la planificació mitjançant llenguatge natural demostra ser efectiva per guiar l'agent cap a objectius útils en entorns simulats rics. CraftyAI sap establir i completar metes intermèdies per progressar.

- L'ús de programes interpretats com a llenguatge d'acció, en comptes de primitives simples, permet a CraftyAI executar plans molt compleixes i seqüenciar grans quantitats d'accions per assolir fites.

- CraftyAI exhibeix algunes limitacions en mantenir un progrés constant i una curiositat sostinguda a molt llarg termini. Un aprenentatge més profund podria millorar la definició d'objectius globals i planificació estratègica.

- Els resultats demostren avantatges clars, evidenciant que l'ús de models de llenguatge a gran escala té un gran potencial per desenvolupar agents artificials cada cop més competents.

- S'ha validat la capacitat de CraftyAI d'adquirir coneixement transferible que li permet completar amb èxit tasques totalment noves, definint un avenç important enfront d'agents limitats a conductes preprogramades.

- Queda obert l'interrogant de fins a quin punt un agent com CraftyAI podria arribar a mostrar intel·ligència i aprenentatge genuí amb entrenament prolongat en mons virtuals rics, constituint una àrea d'investigació emocionant.

Els experiments realitzats han pogut caracteritzar les fortaleses i limitacions actuals de CraftyAI, alhora que s'entreveuen les enormes possibilitats que obre de cara a desenvolupar agents artificials cada cop més versàtils i competents mitjançant l'aprenentatge profund en entorns simulats.

A continuació es pot veure el resultat en aquest vídeo.

## 8. CONCLUSIONS

En aquest treball s'ha fet una investigació profunda sobre la possibilitat de desenvolupar agents virtuals autònoms mitjançant les tècniques més innovadores de la intel·ligència artificial.

La hipòtesi planteada era si seria viable, amb la tecnologia actual, crear agents intel·ligents complets capaços d'aprendre i prendre decisions de forma totalment independent en entorns dinàmics i complexos com els videojocs.

Concretament, es plantejava determinar la viabilitat tècnica i social d'assolir agents autònoms competents en el futur pròxim, almenys en dominis específics acotats. Per abordar aquest interrogant, es va formular la hipòtesi que seria possible desenvolupar un agent virtual autònom que aprengués i actués completament sol en l'entorn Minecraft.

Els resultats experimentals obtinguts després de la implementació i proves exhaustives de l'agent CraftyAI en aquest entorn permeten validar completament la hipòtesi plantejada.

CraftyAI ha exhibit totes les capacitats esperables d'un agent autònom en un entorn complex com Minecraft: exploració proactiva, curiositat sostinguda, recol·lecció de recursos, fabricació d'eines, construcció d'estructures, supervivència, lluita i moltes altres.

En tots els experiments, l'agent ha funcionat de forma totalment autònoma sense cap mena de supervisió, guia o recompenses externes. Ha mostrat comportaments d'aprenentatge, optimitzant les seves polítiques d'acció i exhibint fites cada cop més ambicioses.

Per exemple, en les primeres iteracions CraftyAI es limitava a recollir fusta i roques a l'abast i explorar l'entorn proper. No obstant això, ràpidament va aprendre a fabricar eines de pedra, localitzar i extreure minerals valuosos, construir refugis complexos,

caçar animals, conrear cultius i, en general, actuar de manera competent i flexible per sobreviure i prosperar en qualsevol situació.

A escala tècnica, s'ha demostrat que és perfectament viable implementar agents autònoms com CraftyAI fent servir eines d'aprenentatge profund, processament de llenguatge natural i cerca heurística. La combinació d'aquestes tècniques permet dotar als agents d'habilitats generalistes que transfereixen bé entre dominis.

Els resultats obtinguts permeten validar que, en entorns acotats com els videojocs, ja és possible assolir agents artificials amb un alt grau d'autonomia utilitzant les tecnologies de la intel·ligència artificial més recents. Després de diverses partides, CraftyAI ha evidenciat la seva competència també en entorns complexes i hostils canviant el seu mode de joc de Pacífic a Fàcil, matant a zombis i intentant sobreviure la nit amb èxit.

Tanmateix, també s'han identificat algunes limitacions que constitueixen oportunitats per a investigacions futures. En alguns escenaris molt complexos, CraftyAI tenia dificultats per mantenir un progrés constant a molt llarg termini. També, gràcies al temps de resposta lent de part de GPT-4, la interacció de CraftyAI amb humans posa de manifest que té dificultats de velocitat de computació i resposta en temps real.

També s'han complert tots els objectius plantejats, ja que:

- El disseny de l'agent intel·ligent ha estat basat en xarxes neuronals profundes i ha estat capaç d'actuar de manera autònoma.
- S'han fet servir llenguatges de programació i frameworks d'aprenentatge profund com PyTorch o TensorFlow.
- S'ha entrenat l'agent en un entorn simulat de Minecraft fent servir algoritmes de reforç.
- S'ha avaluat el comportament de l'agent en diversos escenaris del joc, tot analitzant la seva capacitat d'aprendre i prendre decisions de forma autònoma, tal com s'ha explicat.

Seria interessant explorar mecanismes de motivació intrínseca i recompenses internes per fomentar la curiositat i l'establiment d'objectius abstractes per part de l'agent.

També quedaria pendent provar l'enfocament en altres videojocs i, eventualment, en simulacions del món real més properes a aplicacions pràctiques. Igualment, es podrien explorar altres tècniques d'aprenentatge automàtic i representacions de l'agent per millorar l'escalabilitat.

Més enllà dels avenços concrets assolits, aquest treball ha obert diverses qüestions fonamentals de cara al futur. La possibilitat certa de desenvolupar agents autònoms planteja interrogants ètics i socials sobre el paper de la intel·ligència artificial en la nostra societat.

Es requerirà precaució i una regulació adequada per garantir que els futurs agents autònoms s'alineen amb els valors humans i no representin una amenaça. També caldrà estudiar l'impacte econòmic i social d'aquesta tecnologia per mitigar efectes negatius.

En conclusió, aquesta investigació ha permès assolir amb èxit els objectius establerts, validant que és viable crear agents virtuals autònoms competents en entorns complexos com els videojocs mitjançant les tècniques de la intel·ligència artificial més punteres.

Els resultats obren moltes línies de continuïtat cap a agents cada cop més versàtils i amb major grau d'autonomia. Alhora, posen de manifest la necessitat d'un debat social responsable sobre l'adopció d'aquesta tecnologia revolucionària.

Queda un futur apassionant per explorar tot el potencial transformador de la intel·ligència artificial autònoma, però sempre sota un prisma d'ètica, control democràtic i progressivitat responsable.

## 9. AGRAÏMENTS

Vull expressar el meu sincer agraïment a totes les persones que han contribuït d'una manera o altra a fer possible aquest treball de recerca.

En primer lloc, dono les gràcies a la meva tutora, la professora Adoración Cañal, per la seva dedicació, paciència i savis consells al llarg de tot el procés d'elaboració del treball. Les seves orientacions i feedback constant han estat indispensables per encarrilar i enriquir la investigació.

També vull mostrar el meu agraïment totes les persones que han col·laborat per ajudar a la creació de CraftyAI i per compartir el seu expert coneixement en intel·ligència artificial i aprenentatge de màquina. Les seves explicacions i recomanacions de recursos han ampliat enormement la meva comprensió d'aquest apassionant camp.

Agraeixo igualment als meus companys i amics, especialment a Valentín Hamed pel seu suport moral durant els moments més intensos del treball i per compartir idees i mostrar interès sincer en la meva recerca.

Finalment, vull expressar la meva gratitud a la meva família, i especialment als meus pares, per confiar sempre en mi i donar-me ànims i força per continuar endavant en els moments de dubtes. Sense el seu amor incondicional, hauria estat impossible completar aquest treball.

A tots vosaltres, moltes gràcies de tot cor. La vostra ajuda ha estat indispensable per dur a bon port aquesta investigació que tanta il·lusió m'ha fet.

## 10. BIBLIOGRAFIA I WEBGRAFIA


Peter Norvig i Stuart J. Russell (2003). *Inteligencia Artificial: Un Enfoque Moderno, segunda edición*. Madrid: Pearson, Prentice Hall

Jordi Torres (2020) *Python Deep Learning, Introducción práctica con Keras y TensorFlow 2.* Barcelona: Marcombo

Erik J. Larson (2022). *El mito de la Inteligencia Artificial, Por qué las máquinas no pueden pensar como nosotros lo hacemos.* Barcelona: Shackleton Books

Lasse Rouhiainen (2018). *Inteligencia artificial. 101 cosas que debes saber hoy sobre nuestro futuro*. Barcelona: PlanetadeLibros

Alpaydin, Ethem (2016). *Machine Learning: The New AI*. MIT Press.

Intel·ligència artificial [en línia]. Wikipedia [Consultat: 23 d'agost de 2023]. Disponible a: https://es.wikipedia.org/wiki/Inteligencia_artificial

Agent intel·ligent [en línia] Wikipedia. [Consultat: 23 d'agost de 2023] Disponible a: https://en.wikipedia.org/wiki/Intelligent_agent

Què és la intel·ligència artificial? [en línia]. Google Cloud, [Consultat: 29 d'agost de 2023]. Disponible a: https://cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419

MineDojo [en línia]. Building Open-Ended Embodied Agents with Internet-Scale Knowledge [Consultat: 11 d'octubre de 2023] Disponible a: https://minedojo.org/

MineRL [en línia]. Buid AI in Minecraft in the BASALT 2022 competition [Consultat: 15 d'octubre de 2023] Disponible a: https://minerl.io/

Mineflayer [en línia]. Mineflayer official [Consultat: 17 d'octubre de 2023] Disponible a: https://prismarinejs.github.io/mineflayer/

Gymnasium [en línia]. Gymnasium documentatium [Consultat: 17 d'octubre de 2023] Disponible a: https://gymnasium.farama.org/

Langchain [en línia]. Expression language makes it easy to create custom chains [Consultat: 17 d'octubre de 2023] Disponible a: https://www.langchain.com/

Python [en línia]. Manual [Consultat: 20 d'octubre de 2023] Disponible a: https:/python.org

Node.js [en línia]. Open source cross platform JavaScript runtime environment [Consultat: 20 d'octubre de 2023] Disponible a: https://nodejs.org

Conda [en línia]. Package, dependency and environment management for any language [Consultat: 20 d'octubre de 2023] Disponible a: https://conda.io

Minecraft [en línia]. Minecraft [Consultat: 22 d'octubre de 2023] Disponible a: https://minecraft.net

# 11. ANNEXOS

## 11.1. ALGORITME DE CRAFTYAI

```
def craftyai (
        environment , # entorn que utilitza codi com a espai d'acció curriculum_agent , # agent del currículum per
        proposar la següent tasca action_agent , # agent d'acció per a la generació de codi
        critic_agent , # agent crític per a l'autoverificació
        skill_manager , # gestor d'habilitats per afegir noves habilitats i la recuperació d'habilitats
) :
        agent_state = environment . reset ()
        while True :
                exploration_progress = (
                        curriculum_agent . get_exploration_progress (
                                                        curriculum_agent . get_completed_tasks () ,
                                curriculum_agent . get_failed_tasks () ,
                        )
                )
                task = curriculum_agent . propose_next_task (
                        agent_state , exploration_progress
                )
                code = None
                environment_feedback = None
                execution_errors = None
                critique = None
                success = False
                # prova com a màxim 4 rondes abans de passar a la següent tasca
                for i in range (4) :
                        skills = skill_manager . retrieve_skills (
                                task , environment_feedback
                        )
                        code = action_agent . generate_code (
                                task ,
                                code ,
                                environment_feedback ,
                                execution_errors ,
                                critique ,
                                skills ,
                        )
                        (
                                agent_state ,
                                environment_feedback ,
                                execution_errors ,
                        ) = environment . step ( code )
                                                        success , critique = critic_agent . check_task_success (
                                task , agent_state
                        )
                        if success :
                                break
                if success :
                        skill_manager . add_skill ( code )
                        curriculum_agent . add_completed_task ( task )
                else :
                        curriculum_agent . add_failed_task ( task )
```

## 11.2. PROMPTING

GPT-4 i GPT-3.5 ofereixen als usuaris la possibilitat de designar la funció de cada missatge d'avís entre tres opcions:

19

- Sistema: Una instrucció d'alt nivell que guia el comportament del model al llarg de la conversa. Estableix el to general i l'objectiu de la interacció.
- Usuari: Una instrucció detallada que guia l'assistent per a la resposta immediata següent.
- Assistent: Un missatge de resposta generat el model.

Vegeu https://platform.openai.com/docs/guides/chat/introduction per a més detalls.

En aquest projecte, es va fer servir GPT4Free (Disponible a: https://g4f.ai) per poder utilitzar els últims models d'IA gratuïtament.

## 11.3. CURRÍCULUM AUTOMÀTIC

### 11.3.1. COMPONENTS EN EL PROMPT

El prompt d'entrada a GPT-4 consta de diversos components:

(1) Directrius que fomenten diversos comportaments i imposen restriccions (perquè la tasca proposada sigui realitzable i verificable): Vegeu la secció 11.3.3 per al prompt complet;

(2) Estat actual de l'agent:

- **Inventari**: Un diccionari d'elements amb recomptes, per exemple, {‘cobblestone’: 4, ‘furnace’: 1, ‘stone_pickaxe’: 1, ‘oak_planks’: 7, ‘dirt’: 6, ‘wooden_pickaxe’: 1, ‘crafting_table’: 1, ‘raw_iron’: 4, ‘coal’: 1}.
- **Equipament:** Armadures o armes equipades pels agents.
- **Blocs propers:** Un conjunt de noms de blocs a una distància de 32 blocs de l'agent, per exemple, ‘dirt’, ‘water’, ‘spruce_planks’, ‘grass_block’, ‘dirt_path’, ‘sugar_cane’, ‘fern’
- **Altres blocs vists recentment:** Blocs que no són a prop ni a l'inventari**.**
- **Entitats properes:** Un conjunt de noms d'entitats a una distància de 32 blocs de l'agent, per exemple 'pig', 'cat', 'villager', 'zombie'.
- **Una llista de cofres vistos per l'agent:** Els cofres són contenidors externs on l'agent pot dipositar objectes. Si un cofre no s'ha obert abans, el contingut és

"Unknown". En cas contrari, es mostren a l'agent els objectes que hi ha dins de cada cofre.

- **Bioma:** Per exemple, ‘plains’, ‘flower_forest’, ‘meadow’, ‘river’, ‘beach’, ‘forest’, ‘snowy_slopes’, ‘frozen_peaks’, ‘old_growth_birch_forest’, ‘ocean’, ‘sunflower_plains’, ‘stony_shore’.
- **Hora**: Una de ‘sunrise’, ‘day’, ‘noon’, ‘sunset’, ‘night’, ‘midnight’.
- **Barres de salut i gana**: El valor màxim és 20;
- **Posició**: Coordenada 3D (x, y, z) de la posició de l'agent al món

(3) Tasques completades i fallides anteriorment.

(4) Context addicional: Vegeu l'apartat 11.3.2.

(5) Cadena de pensament en resposta: Demanem a GPT-4 que primer raoni sobre el progrés actual i després suggereixi la tasca següent.

### 11.3.2. CONTEXT ADDICIONAL

Aprofitem GPT-3.5 per autoformular preguntes que proporcionin un context addicional. Cada pregunta s'aparella amb un concepte que es fa servir per recuperar el document més rellevant de la base de coneixements de la wiki. Introduïm el contingut del document a GPT-3.5 perquè s'autocontestin les preguntes. A la pràctica, l'ús d'una base de coneixements wiki és opcional, ja que GPT-3.5 ja té una bona comprensió de la mecànica del joc Minecraft. Tot i això, la base de coneixement externa resulta avantatjosa si GPT-3.5 no està preentrenat en aquest domini específic. Vegeu la secció 11.3.3 per obtenir el prompt complet.

### 11.3.3. PROMPT COMPLET

Prompt 1: Prompt completa del sistema per al currículum automàtic. La llista de parells pregunta-resposta representa el context addicional (en anglès).

You are a helpful assistant that tells me the next immediate task to do in Minecraft . My ultimate goal is to discover as many diverse things as possible , accomplish as many diverse tasks as possible and become the best Minecraft player in the world .

I will give you the following information :
Question 1: ...
Answer : ...
Question 2: ...
Answer : ...
Question 3: ...
Answer : ...
...
Biome : ...
Time : ...
Nearby blocks : ...
Other blocks that are recently seen : ...
Nearby entities ( nearest to farthest ) : ...
Health : Higher than 15 means I 'm healthy .
Hunger : Higher than 15 means I 'm not hungry .
Position : ...
Equipment : If I have better armor in my inventory , you should ask me to equip it .
Inventory ( xx /36) : ...
Chests : You can ask me to deposit or take items from these chests . There also might be some unknown chest , you should ask me to open and check items inside the unknown chest .
Completed tasks so far : ...
Failed tasks that are too hard : ...
You must follow the following criteria :
1) You should act as a mentor and guide me to the next task based on my current learning progress .
2) Please be very specific about what resources I need to collect , what I need to craft , or what mobs I need to kill .
3) The next task should follow a concise format , such as " Mine [ quantity ] [ block ]" , " Craft [ quantity ] [ item ]" , " Smelt [ quantity ] [ item ]" , " Kill [ quantity ] [ mob ]" , " Cook [ quantity ] [ food ]" , " Equip [ item ]" etc . It should be a single phrase . Do not propose multiple tasks at the same time . Do not mention anything else .
4) The next task should not be too hard since I may not have the necessary resources or have learned enough skills to complete it yet .
5) The next task should be novel and interesting . I should look for rare resources , upgrade my equipment and tools using better materials , and discover new things . I should not be doing the same thing over and over again .
6) I may sometimes need to repeat some tasks if I need to collect more resources to complete more difficult tasks . Only repeat tasks if necessary .
7) Do not ask me to build or dig shelter even if it ' s at night . I want to explore the world and discover new things . I don ' t want to stay in one place .

8) Tasks that require information beyond the player ' s status to verify should be avoided . For instance , " Placing 4 torches " and " Dig a 2x1x2 hole " are not ideal since they require visual confirmation from the screen . All the placing , building , planting , and trading tasks should be avoided . Do not propose task starting with these keywords .

You should only respond in the format as described below :
RESPONSE FORMAT :
Reasoning : Based on the information I listed above , do reasoning about what the next task should be .
Task : The next task .
Here ' s an example response :
Reasoning : The inventory is empty now , chop down a tree to get some wood .
Task : Obtain a wood log .

Prompt 2: Sistema complet per fer preguntes. Proporcionem exemples bons i dolents. (en anglès).

You are a helpful assistant that asks questions to help me decide the next immediate task to do in Minecraft . My ultimate goal is to discover as many things as possible , accomplish as many tasks as possible and become the best Minecraft player in the world .

I will give you the following information :
Biome : ...
Time : ...
Nearby blocks : ...
Other blocks that are recently seen : ...
Nearby entities ( nearest to farthest ) : ...
Health : ...
Hunger : ...
Position : ...
Equipment : ...
Inventory ( xx /36) : ...
Chests : ...
Completed tasks so far : ...
Failed tasks that are too hard : ...

You must follow the following criteria :
1) You should ask at least 5 questions ( but no more than 10 questions ) to help me decide the next immediate task to do . Each question should be followed by the concept that the question is about . 2) Your question should be specific to a concept in Minecraft . Bad example ( the question is too general ) : Question : What is the best way to play Minecraft ?
Concept : unknown
Bad example ( axe is still general , you should specify the type of axe such as wooden axe ) :
What are the benefits of using an axe to gather resources ? Concept : axe
Good example :
Question : How to make a wooden pickaxe ?
Concept : wooden pickaxe
3) Your questions should be self - contained and not require any context .
Bad example ( the question requires the context of my current biome ) : Question : What are the blocks that I can find in my current biome ? Concept : unknown
Bad example ( the question requires the context of my current inventory ) :
Question : What are the resources you need the most currently ? Concept : unknown
Bad example ( the question requires the context of my current inventory ) :
Question : Do you have any gold or emerald resources ?
Concept : gold
Bad example ( the question requires the context of my nearby entities ) :
Question : Can you see any animals nearby that you can kill for food ?
Concept : food
Bad example ( the question requires the context of my nearby blocks ) : Question : Is there any water source nearby ?
Concept : water
Good example :

Question : What are the blocks that I can find in the sparse jungle ?
Concept : sparse jungle

4) Do not ask questions about building tasks ( such as building a shelter ) since they are too hard for me to do .

Let ' s say your current biome is sparse jungle . You can ask questions like :

Question : What are the items that I can find in the sparse jungle ?
Concept : sparse jungle

Question : What are the mobs that I can find in the sparse jungle ?
Concept : sparse jungle

Let ' s say you see a creeper nearby , and you have not defeated a creeper before . You can ask a question like :
Question : How to defeat the creeper ?
Concept : creeper

Let ' s say you last completed task is " Craft a wooden pickaxe ". You can ask a question like :
Question : What are the suggested tasks that I can do after crafting a wooden pickaxe ?

Concept : wooden pickaxe

Here are some more question and concept examples :
Question : What are the ores that I can find in the sparse jungle ? Concept : sparse jungle
( the above concept should not be " ore " because I need to look up the page of "sparse jungle " to find out what ores I can find in the sparse jungle )

Question : How can you obtain food in the sparse jungle
Concept : sparse jungle
( the above concept should not be " food " because I need to look up the page of " sparse jungle " to find out what food I can obtain in the sparse jungle )
Question : How can you use the furnace to upgrade your equipment and make useful items ?
Concept : furnace
Question : How to obtain a diamond ore ?
Concept : diamond ore
Question : What are the benefits of using a stone pickaxe over a wooden pickaxe ?
Concept : stone pickaxe
Question : What are the tools that you can craft using wood planks and sticks ?
Concept : wood planks

You should only respond in the format as described below RESPONSE FORMAT :

Reasoning : ...
Question 1: ...
Concept 1: ...
Question 2: ...
Concept 2: ...
Question 3: ...
Concept 3: ...
Question 4: ...
Concept 4: ...
Question 5: ...
Concept 5: ...
...

Prompt 3: Sistema complet per respondre preguntes. El context representa el contingut opcional d'una base de coneixements. (en anglès).

You are a helpful assistant that answer my question about Minecraft .

I will give you the following information :
Question : ...

You will answer the question based on the context ( only if available and helpful ) and your own knowledge of Minecraft .
1) Start your answer with " Answer : ".
2) Answer " Answer : Unknown " if you don ’ t know the answer .

## 11.4. BIBLIOTECA D'HABILITATS

### 11.4.1. COMPONENTS EN EL PROMPT

El prompt d'entrada a GPT-4 consta dels components següents:
(1) Directrius per a la generació de codi: Vegeu la secció 11.4.2 per conèixer totes les instruccions
(2) APIs primitives de control implementades per nosaltres: Aquestes APIs tenen un doble propòsit: demostren l'ús de les APIs de Mineflayer, i poden ser anomenades directament per GPT-4.

- exploreUntil(bot, direction, maxTime = 60, callback): Permet a l'agent explorar en una adreça fixa durant maxTime. El callback és la condició de parada implementada per l'agent per determinar quan deixar d'explorar
- mineBlock(bot, name, count = 1): Mina i recull el nombre especificat de blocs dins una distància de 32 blocs.
- craftItem(bot, name, count = 1): Crea l'objecte amb una taula de fabricació propera.
- placeItem(bot, name, position): Col·loca el bloc a la posició especificada.
- smeltItem(bot, itemName, fuelName, count = 1): Fon l'objecte amb el combustible especificat. Hi ha d'haver un forn a prop.
- killMob(bot, mobName, timeout = 300): Ataca el mob i recull el seu objecte caigut.
- getItemFromChest(bot, chestPosition, itemsToGet): Es desplaça fins al cofre a la posició especificada i obté objectes del cofre.
- depositItemIntoChest(bot, chestPosition, itemsToDeposit): Es desplaça fins al cofre a la posició especificada i diposita articles al cofre.

(3) API primitives de control proporcionades per Mineflayer:

- await bot.pathfinder.goto(goal): Anar a una posició específica. Tot seguit s'explica com establir l'objectiu.
- new GoalNear(x, y, z, range): Mou el bot a un bloc dins del rang especificat.
- new GoalXZ(x, z): Per a objectius de llarg abast que no tenen un nivell Y específic.
- new GoalGetToBlock(x, y, z): No entrar al bloc, sinó posar-se directament adjacent a ell. Útil per pescar, cultivar, omplir una galleda, i fer servir un llit.

- new GoalFollow(entity, range): Segueix l'entitat especificada dins del rang especificat.
- new GoalPlaceBlock(position, bot.world, {}): Posiciona el bot per col·locar un bloc.
- new GoalLookAtBlock(position, bot.world, {}): Recorregut cap a una posició en què és visible una cara del bloc a la posició.
- bot.isABed(bedBlock): Retorna true si bedBlock és un llit.
- bot.blockAt(position): Retorna el bloc a la posició.
- await bot.equip(item, destination): Equipa l'objecte a la destinació especificada. El destí ha de ser un dels següents: "hand", "head", "tors", "legs", "feet", "off-hand"
- await bot.consume(): Consumeix l'objecte de la mà del bot. Primer ha d'equipar l'objecte a consumir. Útil per menjar, beure pocions, etc
- await bot.fish(): Deixa que el bot pesqui. Abans de trucar a aquesta funció, primer heu d'arribar a un bloc d'aigua i després equipar una canya de pescar. El bot deixarà de pescar automàticament quan atrapi un peix
- await bot.sleep(bedBlock): Dorm fins a l'alba. Primer has d'arribar a un bloc llit.
- await bot.activateBlock(block): Això és el mateix que fer clic amb el botó dret en un bloc al joc. Útil per a botons, portes, etc. Has d'arribar primer al bloc.
- await bot.lookAt(position): Veu la posició especificada. Ha d'acostar a la posició abans de mirar-la. Per omplir una galleda d'aigua, primer l'ha de mirar.
- await bot.activateItem(): Això és el mateix que fer clic amb el botó dret del ratolí per utilitzar l'element a la mà del bot. Útil per fer servir una galleda, etc. Ha d'equipar l'objecte per activar-lo primer.
- await bot.useOn(entity): És el mateix que fer clic amb el botó dret a una entitat del joc. Útil per esquilar una ovella. Primer ha d'arribar a l'entitat.

(4) Habilitats recuperades de la biblioteca d'habilitats.

(5) Codi generat a l'última ronda.

(6) Feedback de l'entorn: El registre del xat al prompt.

(7) Errors d'execució.

(8) Crítica del mòdul d'autoverificació.

(9) Estat actual de l'agent: vegeu l'apartat 11.3.1 per a cada element de l'estat de l'agent.
(10) Tasca proposada pel currículum automàtic.
(11) Context de la tasca: Demanem a GPT-3.5 que sol·liciti suggeriments generals sobre com resoldre la tasca. A la pràctica, se n'encarrega el pla d'estudis automàtic, ja que disposa d'un mecanisme sistemàtic de resposta a preguntes (Sec. 11.3.2).
(12) Indicació de la cadena de pensament en la resposta: Demanem a GPT-4 que primer expliqui la raó per la qual falla el codi de l'última ronda, que després doni plans pas a pas per acabar la tasca i que finalment generi codi. Vegeu l'apartat 11.4.2 per veure el prompt complet.

### 11.4.2. PROMPT COMPLET

Prompt 4: Sistema complet per a la generació de codi (en anglès).

```
You are a helpful assistant that writes Mineflayer javascript code to complete any Minecraft task specified by me .

Here are some useful programs written with Mineflayer APIs .

/*
                Explore until find an iron_ore , use Vec3 (0 , -1 , 0) because iron ores are usually
                    underground
                await exploreUntil ( bot , new Vec3 (0 , -1 , 0) , 60 , () = > { const
                    iron_ore = bot . findBlock ({
                            matching : mcData . blocksByName [" iron_ore "]. id ,
                        maxDistance : 32 ,
                    }) ;
                    return iron_ore ;
                }) ;

                Explore until find a pig , use Vec3 (1 , 0 , 1) because pigs are usually on the surface
                let pig = await exploreUntil ( bot , new Vec3 (1 , 0 , 1) , 60 , () = > { const pig = bot .
                    nearestEntity (( entity ) = > {
                        return (
                            entity . name === " pig " &&
                                entity . position . distanceTo ( bot . entity . position ) < 32
                        ) ;
                    }) ;
                    return pig ;
                }) ;
                */
                async function exploreUntil ( bot , direction , maxTime = 60 , callback ) { /*
                    Implementation of this function is omitted .
                    direction : Vec3 , can only contain value of -1 , 0 or 1 maxTime :
                    number , the max time for exploration
                    callback : function , early stop condition , will be called each second ,
                    exploration will stop if return value is not null

                    Return : null if explore timeout , otherwise return the return value of callback
                    */
                }


                // Mine 3 cobblestone : mineBlock ( bot , " stone " , 3) ;
```

```
async function mineBlock ( bot , name , count = 1) {
        const blocks = bot . findBlocks ({
                matching : ( block ) = > {
                        return block . name === name ;
                } ,
                maxDistance : 32 ,
                count : count ,
        }) ;
        const targets = [];
        for ( let i = 0; i < Math . min ( blocks . length , count ) ; i ++) { targets .
                push ( bot . blockAt ( blocks [ i ]) ) ;
        }
        await bot . collectBlock . collect ( targets , { ignoreNoPath : true }) ; }


// Craft 8 oak_planks from 2 oak_log ( do the recipe 2 times ) : craftItem ( bot ,
       " oak_planks " , 2) ;



// You must place a crafting table before calling this function async function
craftItem ( bot , name , count = 1) {
        const item = mcData . itemsByName [ name ];
        const craftingTable = bot . findBlock ({
                    matching : mcData . blocksByName . crafting_table . id ,
                maxDistance : 32 ,
        }) ;
        await bot . pathfinder . goto (
                new GoalLookAtBlock ( craftingTable . position , bot . world ) ) ;
        const recipe = bot . recipesFor ( item . id , null , 1 , craftingTable ) [0]; await bot . craft
        ( recipe , count , craftingTable ) ;
}


// Place a crafting_table near the player , Vec3 (1 , 0 , 0) is just an example , you
shouldn ' t always use that : placeItem ( bot , " crafting_table " , bot . entity . position .
offset (1 , 0 , 0) ) ; async function placeItem ( bot , name , position ) {
        const item = bot . inventory . findInventoryItem ( mcData . itemsByName [ name ].
       id ) ;
        // find a reference block
        const faceVectors = [
                new Vec3 (0 , 1 , 0) ,
                new Vec3 (0 , -1 , 0) ,
                new Vec3 (1 , 0 , 0) ,
                new Vec3 ( -1 , 0 , 0) ,
                new Vec3 (0 , 0 , 1) ,
                new Vec3 (0 , 0 , -1) ,
        ];
        let referenceBlock = null ;
        let faceVector = null ;
        for ( const vector of faceVectors ) {
                        const block = bot . blockAt ( position . minus ( vector ) ) ;
                if ( block ?. name !== " air ") {
                        referenceBlock = block ;
                        faceVector = vector ;
                        break ;
                }
        }
        // You must first go to the block position you want to place await bot . pathfinder .
       goto ( new GoalPlaceBlock ( position , bot . world , {}) ) ;
        // You must equip the item right before calling placeBlock await bot .
        equip ( item , " hand ") ;
                await bot . placeBlock ( referenceBlock , faceVector ) ;
}


// Smelt 1 raw_iron into 1 iron_ingot using 1 oak_planks as fuel : smeltItem ( bot , "
       raw_iron " , " oak_planks ") ;
// You must place a furnace before calling this function async function smeltItem
( bot , itemName , fuelName , count = 1) { const item = mcData . itemsByName
[ itemName ];
```

```
        const fuel = mcData . itemsByName [ fuelName ];
        const furnaceBlock = bot . findBlock ({
            matching : mcData . blocksByName . furnace . id ,
              maxDistance : 32 ,
        }) ;
        await bot . pathfinder . goto (
              new GoalLookAtBlock ( furnaceBlock . position , bot . world ) ) ;
               const furnace = await bot . openFurnace ( furnaceBlock ) ;
        for ( let i = 0; i < count ; i ++) {
                    await furnace . putFuel ( fuel . id , null , 1) ;



                    await furnace . putInput ( item . id , null , 1) ;
                              // Wait 12 seconds for the furnace to smelt the item
              await bot . waitForTicks (12 * 20) ;
              await furnace . takeOutput () ;
        }
        await furnace . close () ;
}


// Kill a pig and collect the dropped item : killMob ( bot , " pig " , 300) ; async function
killMob ( bot , mobName , timeout = 300) { const entity = bot . nearestEntity (
              ( entity ) = >
                    entity . name === mobName &&
                              entity . position . distanceTo ( bot . entity . position ) < 32
        ) ;
        await bot . pvp . attack ( entity ) ;
        await bot . pathfinder . goto (
              new GoalBlock ( entity . position .x , entity . position .y , entity . position . z )
        ) ;
}


// Get a torch from chest at (30 , 65 , 100) : getItemFromChest ( bot , new Vec3 (30 , 65 ,
      100) , {" torch ": 1}) ;
// This function will work no matter how far the bot is from the chest .
async function getItemFromChest ( bot , chestPosition , itemsToGet ) { await
        moveToChest ( bot , chestPosition ) ;
            const chestBlock = bot . blockAt ( chestPosition ) ;
                 const chest = await bot . openContainer ( chestBlock ) ;
        for ( const name in itemsToGet ) {
              const itemByName = mcData . itemsByName [ name ];
                    const item = chest . findContainerItem ( itemByName . id ) ;
              await chest . withdraw ( item . type , null , itemsToGet [ name ]) ; }
        await closeChest ( bot , chestBlock ) ;
}
// Deposit a torch into chest at (30 , 65 , 100) : depositItemIntoChest ( bot , new Vec3 (30 ,
      65 , 100) , {" torch ": 1}) ;
// This function will work no matter how far the bot is from the chest .
async function depositItemIntoChest ( bot , chestPosition , itemsToDeposit ) {
        await moveToChest ( bot , chestPosition ) ;
            const chestBlock = bot . blockAt ( chestPosition ) ;
                 const chest = await bot . openContainer ( chestBlock ) ;
        for ( const name in itemsToDeposit ) {
              const itemByName = mcData . itemsByName [ name ];
                const item = bot . inventory . findInventoryItem ( itemByName . id ) ; await
        chest . deposit ( item . type , null , itemsToDeposit [ name ]) ; }
        await closeChest ( bot , chestBlock ) ;
}
// Check the items inside the chest at (30 , 65 , 100) :
       checkItemInsideChest ( bot , new Vec3 (30 , 65 , 100) ) ;
// You only need to call this function once without any action to finish task of
      checking items inside the chest .
async function checkItemInsideChest ( bot , chestPosition ) { await
        moveToChest ( bot , chestPosition ) ;
            const chestBlock = bot . blockAt ( chestPosition ) ;
        await bot . openContainer ( chestBlock ) ;
        // You must close the chest after opening it if you are asked to open a chest
        await closeChest ( bot , chestBlock ) ;
```

}

await bot . pathfinder . goto ( goal ) ; // A very useful function . This function may change your main - hand equipment .
// Following are some Goals you can use :
new GoalNear (x , y , z , range ) ; // Move the bot to a block within the specified range of the specified block . ‘x ‘ , ‘y ‘ , ‘z ‘ , and ‘ range ‘ are ‘ number ‘
new GoalXZ (x , z ) ; // Useful for long - range goals that don ’ t have a specific Y level . ‘x ‘ and ‘z ‘ are ‘ number ‘
new GoalGetToBlock (x , y , z ) ; // Not get into the block , but get directly adjacent to it . Useful for fishing , farming , filling bucket , and beds . ‘x ‘, ‘y ‘ , and ‘z ‘ are ‘ number ‘
new GoalFollow ( entity , range ) ; // Follow the specified entity within the specified range . ‘ entity ‘ is ‘ Entity ‘ , ‘ range ‘ is ‘ number ‘ new GoalPlaceBlock ( position , bot . world , {}) ; // Position the bot in order to place a block . ‘ position ‘ is ‘ Vec3 ‘
new GoalLookAtBlock ( position , bot . world , {}) ; // Path into a position where a blockface of the block at position is visible . ‘ position ‘ is ‘ Vec3 ‘

// These are other Mineflayer functions you can use :
bot . isABed ( bedBlock ) ; // Return true if ‘ bedBlock ‘ is a bed bot . blockAt ( position ) ;
// Return the block at ‘ position ‘. ‘ position ‘ is ‘ Vec3 ‘

// These are other Mineflayer async functions you can use : await bot . equip ( item , destination ) ; // Equip the item in the specified destination . ‘ item ‘ is ‘ Item ‘ , ‘ destination ‘ can only be " hand " , " head " , " torso " , " legs " , " feet " , " off - hand "
await bot . consume () ; // Consume the item in the bot ’ s hand . You must equip the item to consume first . Useful for eating food , drinking potions , etc .
await bot . fish () ; // Let bot fish . Before calling this function , you must first get to a water block and then equip a fishing rod . The bot will automatically stop fishing when it catches a fish
await bot . sleep ( bedBlock ) ; // Sleep until sunrise . You must get to a bed block first
await bot . activateBlock ( block ) ; // This is the same as right - clicking a block in the game . Useful for buttons , doors , using hoes , etc . You must get to the block first
await bot . lookAt ( position ) ; // Look at the specified position . You must go near the position before you look at it . To fill bucket with water , you must lookAt first . ‘ position ‘ is ‘ Vec3 ‘
await bot . activateItem () ; // This is the same as right - clicking to use the item in the bot ’ s hand . Useful for using buckets , etc . You must equip the item to activate first
await bot . useOn ( entity ) ; // This is the same as right - clicking an entity in the game . Useful for shearing sheep , equipping harnesses , etc . You must get to the entity first

{ retrieved_skills }

At each round of conversation , I will give you
Code from the last round : ...
Execution error : ...
Chat log : ...
Biome : ...
Time : ...
Nearby blocks : ...
Nearby entities ( nearest to farthest ) :
Health : ...
Hunger : ...
Position : ...
Equipment : ...
Inventory ( xx /36) : ...
Chests : ...
Task : ...
Context : ...
Critique : ...

You should then respond to me with
Explain ( if applicable ) : Are there any steps missing in your plan ? Why does the code not complete the task ? What does the chat log and execution error imply ?
Plan : How to complete the task step by step . You should pay attention to Inventory since it tells what you have . The task completeness check is also based on your final inventory .
Code :
1) Write an async function taking the bot as the only argument . 2) Reuse the above useful programs as much as possible . - Use ‘ mineBlock ( bot , name , count ) ‘ to collect blocks . Do not use ‘ bot . dig ‘ directly .

- Use ‘ craftItem ( bot , name , count ) ‘ to craft items . Do not use ‘ bot . craft ‘ directly .
- Use ‘ smeltItem ( bot , name count ) ‘ to smelt items . Do not use ‘ bot . openFurnace ‘ directly .
- Use ‘ placeItem ( bot , name , position ) ‘ to place blocks . Do not use ‘ bot . placeBlock ‘ directly .
- Use ‘ killMob ( bot , name , timeout ) ‘ to kill mobs . Do not use ‘ bot . attack ‘ directly .

3) Your function will be reused for building more complex functions . Therefore , you should make it generic and reusable . You should not make strong assumption about the inventory ( as it may be changed at a later time ) , and therefore you should always check whether you have the required items before using them . If not , you should first collect the required items and reuse the above useful programs .

4) Functions in the " Code from the last round " section will not be saved or executed . Do not reuse functions listed there . 5) Anything defined outside a function will be ignored , define all your variables inside your functions .

6) Call ‘ bot . chat ‘ to show the intermediate progress . 7) Use ‘ exploreUntil ( bot , direction , maxDistance , callback ) ‘ when you cannot find something . You should frequently call this before mining blocks or killing mobs . You should select a direction at random every time instead of constantly using (1 , 0 , 1) . 8) ‘ maxDistance ‘ should always be 32 for ‘ bot . findBlocks ‘ and ‘ bot . findBlock ‘. Do not cheat .

9) Do not write infinite loops or recursive functions . 10) Do not use ‘ bot . on ‘ or ‘ bot . once ‘ to register event listeners . You definitely do not need them .

11) Name your function in a meaningful way ( can infer the task from the name ) .

You should only respond in the format as described below : RESPONSE FORMAT :

Explain : ...

Plan :

1) ...

2) ...

3) ...

...

Code :

```javascript
// helper functions ( only if needed , try to avoid them ) ...
// main function after the helper functions
async function yourMainFunctionName ( bot ) {
    // ...
}
```

Prompt 5: Prompt complet del sistema per generar descripcions de funcions. S'utilitza quan s'afegeix una habilitat nova a la biblioteca d'habilitats. Donem un exemple al prompt. (en anglès).

```
You are a helpful assistant that writes a description of the given function written in Mineflayer javascript
      code .

1) Do not mention the function name .
2) Do not mention anything about ‘ bot . chat ‘ or helper functions . 3) There might be some helper functions
before the main function , but you only need to describe the main function .
4) Try to summarize the function in no more than 6 sentences . 5) Your response should be a
single line of text .

For example , if the function is :

async function mineCobblestone ( bot ) {
    // Check if the wooden pickaxe is in the inventory , if not , craft one
    let woodenPickaxe = bot . inventory . findInventoryItem ( mcData . itemsByName ["
       wooden_pickaxe "]. id ) ;
    if (! woodenPickaxe ) {
        bot . chat (" Crafting a wooden pickaxe .") ;
        await craftWoodenPickaxe ( bot ) ;
        woodenPickaxe = bot . inventory . findInventoryItem ( mcData . itemsByName [" wooden_pickaxe "]. id ) ;
    }

    // Equip the wooden pickaxe if it exists
    if ( woodenPickaxe ) {
        await bot . equip ( woodenPickaxe , " hand ") ;

        // Explore until we find a stone block
        await exploreUntil ( bot , new Vec3 (1 , -1 , 1) , 60 , () = > { const stone = bot . findBlock
            ({
                                matching : mcData . blocksByName [" stone "]. id ,
                                maxDistance : 32
                            }) ;
                            if ( stone ) {
                                return true ;
                            }
                        }) ;

                                    // Mine 8 cobblestone blocks using the wooden pickaxe
                        bot . chat (" Found a stone block . Mining 8 cobblestone blocks .") ; await
                        mineBlock ( bot , " stone ", 8) ;
                                    bot . chat (" Successfully mined 8 cobblestone blocks .") ;

                        // Save the event of mining 8 cobblestone
                        bot . save (" cobblestone_mined ") ;
                    } else {
                        bot . chat (" Failed to craft a wooden pickaxe . Cannot mine
                        cobblestone .") ;
                    }
                }

                The main function is ‘ mineCobblestone ‘.

                Then you would write :
                The function is about mining 8 cobblestones using a wooden pickaxe . First check if a
                      wooden pickaxe is in the inventory . If not , craft one . If the wooden pickaxe is
                      available , equip the wooden pickaxe in the hand . Next , explore the environment
                      until finding a stone block . Once a stone block is found , mine a total of 8 cobblestone
                      blocks using the wooden pickaxe
```

### 11.4.3. EXEMPLES

Exemple 1 de la biblioteca d'habilitats: craftWoodenPlanks.

```
async function craftWoodenPlanks ( bot ) {
    const logNames = [" oak_log " , " birch_log " , " spruce_log " , " jungle_log " , " acacia_log " , " dark_oak_log " , "
        mangrove_log " ];
    const plankNames = [" oak_planks " , " birch_planks " , " spruce_planks " , " jungle_planks " , " acacia_planks " ,
        " dark_oak_planks " , "
        mangrove_planks " ];
    const logInInventory = logNames . find ( logName = > bot . inventory . count ( mcData . itemsByName [
        logName ]. id ) > 0) ;
    if (! logInInventory ) {
        bot . chat (" No wooden log in inventory . Mining a wooden log ... " ) ; await mineWoodLog ( bot ) ;
    }
    const logIndex = logNames . indexOf ( logInInventory ) ;
    const plankName = plankNames [ logIndex ];
    bot . chat ( ' Crafting 4 ${ plankName }... ') ;
    await craftItem ( bot , plankName , 1) ;
    bot . chat ( '4 $ { plankName } crafted . ') ;
}
```

Exemple de biblioteca d'habilitats 2: mineTenCobbledDeepslateBelowY0.

```
async function mineTenCobbledDeepslateBelowY0 ( bot ) {
    // Equip the iron pickaxe
    const ironPickaxe = bot . inventory . findInventoryItem ( mcData . itemsByName [" iron_pickaxe " ].
        id ) ;
    await bot . equip ( ironPickaxe , " hand ") ;

    // Find cobbled_deepslate blocks below Y =0
    const cobbledDeepslateBlocks = await exploreUntil ( bot , new Vec3 (1 , -1 , 1) , 60 , () = > {
        const cobbledDeepslate = bot . findBlock ({
            matching : mcData . blocksByName [ " cobbled_deepslate " ]. id , maxDistance : 32 ,
            position : pos = > pos . y < 0
        }) ;
        return cobbledDeepslate ;
    }) ;

    // Mine 10 cobbled_deepslate blocks
    await mineBlock ( bot , " cobbled_deepslate " , 10) ;
    bot . chat (" 10 cobbled_deepslate mined below Y =0. " ) ;
}
```

Exemple de biblioteca d'habilitats 3: smeltFiveRawIronV2.

```
    async function findSuitablePosition ( bot ) {
        const offsets = [ new Vec3 (1 , 0 , 0) , new Vec3 ( -1 , 0 , 0) , new Vec3 (0 , 0 , 1) , new Vec3 (0 , 0 , -1) ];
        for ( const offset of offsets ) {
            const position = bot . entity . position . offset ( offset .x , offset .y , offset . z ) ;
            const block = bot . blockAt ( position ) ;
            if ( block . name === " air ") {
                return position ;
            }
        }
        return null ;
    }

    async function smeltFiveRawIron ( bot ) {
        // Check if there is a furnace in the inventory
        const furnaceItem = bot . inventory . findInventoryItem ( mcData . itemsByName . furnace . id )
```

```
    ;

// If not , craft a furnace using the available cobblestone if (! furnaceItem ) {
    await craftFurnace ( bot ) ;
}

// Find a suitable position to place the furnace
const furnacePosition = await findSuitablePosition ( bot ) ; if (! furnacePosition ) {
    bot . chat (" Could not find a suitable position to place the furnace . ") ;
    return ;
}

// Place the furnace at the suitable position
await placeItem ( bot , " furnace " , furnacePosition ) ;

                                        // Smelt 5 raw iron using the available coal as fuel
                    await smeltItem ( bot , " raw_iron " , " coal ", 5) ;
                    bot . chat (" 5 raw iron smelted . ") ;
                }
```

Exemple 4 de la biblioteca d'habilitats: fillBucketWithWater.

```
async function fillBucketWithWater ( bot ) {
// Find a water block nearby
    const waterBlock = await exploreUntil ( bot , new Vec3 (1 , 0 , 1) , 60 , () = > {
        const water = bot . findBlock ({
            matching : mcData . blocksByName . water . id ,
            maxDistance : 32
        }) ;
        return water ;
    }) ;
    if (! waterBlock ) {
        bot . chat (" Could not find water . ") ;
        return ;
    }

    // Go to the water block
    const adjacentBlock = waterBlock . position . offset (0 , 1 , 0) ; await bot . pathfinder . goto (
      new GoalGetToBlock ( adjacentBlock .x , adjacentBlock .y , adjacentBlock . z )
      ) ;

      // Look at the water block
    await bot . lookAt ( waterBlock . position , true ) ;

    // Equip the bucket
    const bucket = bot . inventory . findInventoryItem ( mcData . itemsByName . bucket . id ) ;
    await bot . equip ( bucket , " hand ") ;
    // Activate the bucket to collect water
    await bot . activateItem () ;
    bot.chat(" Filled the bucket with water ." ) ;
    }
```

Exemple 5 de la biblioteca d'habilitats: catchFiveFishSafely.

```
async function catchFiveFishSafely ( bot ) {
    // Check if the bot has a fishing rod in its inventory let fishingRod = bot . inventory . findInventoryItem ( mcData .
    itemsByName . fishing_rod . id ) ;
    if (! fishingRod ) {
        await craftFishingRod ( bot ) ;
        fishingRod = bot . inventory . findInventoryItem ( mcData . itemsByName . fishing_rod . id ) ;
    }

    // Find a nearby water block
    let waterBlock ;
    while (! waterBlock ) {
        waterBlock = await exploreUntil ( bot , new Vec3 (1 , 0 , 1) , 60 , () = > {
            const foundWaterBlock = bot . findBlock ({
                matching : mcData . blocksByName . water . id ,
                maxDistance : 32
            }) ;
            return foundWaterBlock ;
        }) ;
        if (! waterBlock ) {
            bot . chat (" No path to the water block . Trying to find another water block ... ") ;
        }
    }

    // Move to a block adjacent to the water block
    const adjacentBlock = waterBlock . position . offset (0 , 1 , 0) ; await bot . pathfinder . goto ( new
    GoalBlock ( adjacentBlock .x , adjacentBlock .y , adjacentBlock . z ) ) ;

    // Look at the water block
    await bot . lookAt ( waterBlock . position ) ;

    // Equip the fishing rod
    await bot . equip ( fishingRod , " hand ") ;

    // Fish in the water 5 times
    for ( let i = 0; i < 5; i ++) {
        try {
            await bot . fish () ;
            bot . chat ( ‘ Fish $ { i + 1} caught . ‘) ;
        } catch ( error ) {
            if ( error . message === " Fishing cancelled ") {
                bot . chat (" Fishing was cancelled . Trying again ... " ) ;
                i - -; // Retry the same iteration
            } else {
                throw error ;
            }
        }
    }
}
```

## 11.5. AUTOVERIFICACIÓ

### 11.5.1. COMPONENTS EN EL PROMPT

El prompt d'entrada a GPT-4 consta dels components següents:
(1) L'estat de l'agent: Excloem de l'estat de l'agent altres blocs vistos recentment i les entitats properes, ja que no són útils per avaluar si la tasca és completa. Vegeu la secció 11.3.1 per a cada element de l'estat de l'agent;
(2) Tasca proposada pel currículum automàtic;
(3) Context de la tasca: Demanem a GPT-3.5 que sol·liciti suggeriments generals sobre com resoldre la tasca. A la pràctica, se n'encarrega el pla d'estudis automàtic, ja que disposa d'un mecanisme sistemàtic de resposta a preguntes (Sec. 11.3.2);
(4) Indicació de la cadena de pensament en la resposta: Demanem a GPT-4 que raoni inicialment sobre l'èxit o el fracàs de la tasca, que emeti una variable booleana que indiqui el resultat de la tasca i que, finalment, proporcioni una crítica a l'agent si la tasca fracassa.
(5) Exemples d'aprenentatge en context.

### 11.5.2. PROMPT COMPLET

Prompt 6: Prompt complet del sistema per a autoverificació.

```
You are an assistant that assesses my progress of playing Minecraft and provides useful guidance .

You are required to evaluate if I have met the task requirements . Exceeding the task requirements is also
    considered a success while failing to meet them requires you to provide critique to help me improve .

I will give you the following information :

Biome : The biome after the task execution .
Time : The current time .
Nearby blocks : The surrounding blocks . These blocks are not collected yet . However , this is useful for
some placing or planting tasks . Health : My current health .
Hunger : My current hunger level . For eating task , if my hunger level is 20.0 , then I successfully ate the
    food .
Position : My current position .
Equipment : My final equipment . For crafting tasks , I sometimes equip the crafted item .
Inventory ( xx /36) : My final inventory . For mining and smelting tasks , you only need to check inventory .
Chests : If the task requires me to place items in a chest , you can find chest information here .
Task : The objective I need to accomplish .
Context : The context of the task .

You should only respond in JSON format as described below : {
    " reasoning ": " reasoning " ,
    " success ": boolean ,
    " critique ": " critique " ,
}
Ensure the response can be parsed by Python ‘ json . loads ‘ , e . g .: no trailing commas , no single
    quotes , etc .

Here are some examples :
```

INPUT :
Inventory (2/36) : { ’ oak_log ’:2 , ’ spruce_log ’:2}
Task : Mine 3 wood logs

RESPONSE :
{
" reasoning ": " You need to mine 3 wood logs . You have 2 oak logs and 2 spruce logs , which add up to 4 wood logs ." ,
" success ": true ,
" critique ": ""
}

INPUT :
Inventory (3/36) : { ’ crafting_table ’: 1 , ’ spruce_planks ’: 6 , ’ stick ’: 4}

Task : Craft a wooden pickaxe

RESPONSE :
{
" reasoning ": " You have enough materials to craft a wooden pickaxe , but you didn ’ t craft it ." ,
" success ": false ,
" critique ": " Craft a wooden pickaxe with a crafting table using 3 spruce planks and 2 sticks ."
}

INPUT :
Inventory (2/36) : { ’ raw_iron ’: 5 , ’ stone_pickaxe ’: 1} Task : Mine 5 iron_ore

RESPONSE :
{
" reasoning ": " Mining iron_ore in Minecraft will get raw_iron . You have 5 raw_iron in your inventory ." ,
" success ": true ,
" critique ": ""
}

INPUT :
Biome : plains

Nearby blocks : stone , dirt , grass_block , grass , farmland , wheat Inventory (26/36) : ...

Task : Plant 1 wheat seed .

RESPONSE :
{
" reasoning ": " For planting tasks , inventory information is useless . In nearby blocks , there is farmland and wheat , which means you succeed to plant the wheat seed ." ,
" success ": true ,
" critique ": ""
}

INPUT :
Inventory (11/36) : {... ,’ rotten_flesh ’: 1}

Task : Kill 1 zombie

Context : ...

RESPONSE
{
" reasoning ": " You have rotten flesh in your inventory , which means you successfully killed one zombie ." ,
" success ": true ,
" critique ": ""
}

INPUT :
Hunger : 20.0/20.0

Inventory (11/36) : ...

Task : Eat 1 ...

Context : ...

RESPONSE
{
" reasoning ": " For all eating task , if the player ' s hunger is 20.0 , then the player successfully ate the food ." ,
" success ": true ,
" critique ": ""
}

INPUT :
Nearby blocks : chest

Inventory (28/36) : { ' rail ': 1 , ' coal ': 2 , ' oak_planks ': 13 , ' copper_block ': 1 , ' diorite ': 7 , ' cooked_beef ': 4 , ' granite ': 22 , ' cobbled_deepslate ': 23 , ' feather ': 4 , ' leather ': 2 , ' cooked_chicken ': 3 , ' white_wool ': 2 , ' stick ': 3 , ' black_wool ': 1 , ' stone_sword ': 2 , ' stone_hoe ': 1 , ' stone_axe ': 2 , ' stone_shovel ': 2 , ' cooked_mutton ': 4 , ' cobblestone_wall ': 18 , ' crafting_table ': 1 , ' furnace ': 1 , ' iron_pickaxe ': 1 , ' stone_pickaxe ': 1 , ' raw_copper ': 12}

Chests :
(81 , 131 , 16) : { ' andesite ': 2 , ' dirt ': 2 , ' cobblestone ': 75 , ' wooden_pickaxe ': 1 , ' wooden_sword ': 1}

Task : Deposit useless items into the chest at (81 , 131 , 16) Context : ...

RESPONSE
{
" reasoning ": " You have 28 items in your inventory after depositing , which is more than 20. You need to deposit more items from your inventory to the chest ." ,
" success ": false ,
" critique ": " Deposit more useless items such as copper_block , diorite , granite , cobbled_deepslate , feather , and leather to meet the requirement of having only 20 occupied slots in your inventory ."
}

## 11.6. CODI COMPLET

Tot el codi de CraftyAI es troba disponible de manera oberta en GitHub a:

https://github.com/TheUnrealZaka/CraftyAI