eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Lana

Testu-jokoetarako errefortzu bidezko ikasketa

Egilea

Garikoitz Marcos Cruchaga

2020

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Lana

Testu-jokoetarako errefortzu bidezko ikasketa

Egilea

Garikoitz Marcos Cruchaga

Zuzendariak

Eneko Agirre, Gorka Azkune

Esker onak

Lehenik eta behin, eskerrak eman nahi dizkiet nire zuzendariei, Eneko eta Gorka, proiektu osoan zehar nirekin egoteagatik eta izandako edozein arazorekin laguntzeagatik.

Aita ta amari, gradua hasi nuenetik erakutsitako babesagatik.

Pisukideei, izandako esperientzia ezin hobeagatik.

Lagunei.

Kuadrillari, haien interesa eta pazientziagatik. Eskerrik asko, Nortzuk.

Laburpena

Proiektu honetan errefortzu bidezko ikasketa erabili nahi da lengoaia naturalen ikasketa garatzeko. Horretarako, testu-jokoak erabili dira ikasketa paradigma honetan sakontzeko. Testu-jokoak espazio konbinatorial eta konposizionalak bezala interpreta daitezke, beraz, ingurune ezin hobeak dira lengoaia naturalaren ikasketa gauzatzeko. Proiektuaren helburua erabiliko den sistemak TextWorld (Côté et al., 2018) ingurunean errealitatea eta hizkuntzaren arteko zati konkretu bat RL bidez ikasi duen ala ez aztertzea izan da. Ondorioei dagokionez, esperimentazioa egin eta gero, sistema munduko topologia ikasteko gai ez dela ikusi da, beste mundu-errepresentazio sistema baten beharra agerian utziz.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Lengoaia Naturalaren Prozesamenduan (LNP) oso ohikoa da ikasketa gainbegiratua erabiltzea. Proiektu honetan beste bide bat ikertu nahi da, errefortzu bidezko ikasketa (RL) lengoaia naturala ikasteko. Horretarako, testuan oinarritutako jokoak erabili dira agenteak partzialki ikusi daiteken mundu batean nola aritzen diren ikusteko.

Testuan oinarritutako jokoak, *Interactive Fiction (IF)* ere deituak, simulazio interaktiboak dira, non jokoaren egoera eta ekintzak testu bidez deskribatzen diren, esaterako, ***go north*** edota ***open door with key***.

Mota honetako jokoak 80ko hamarkadan hasi ziren garrantzia lortzen, batez ere, *Zork I* (Blank and Lebling, 1980) jokoa merkaturatu zenean. Izan ere, egun, joko garrantzitsuenetarikotzat hartzen da eta garatzen diren hainbat agentek (Ammanabrolu and Hausknecht, 2020; Hausknecht et al., 2019) erabiltzen dute haien sistemak entrenatu eta testatzeko. Esan beharrekoa da joko hauen zailtasun maila dezente handia dela, hots, Hausknecht et al.-ek (2019) garatutako agenteak testu-joko multzo bateko (Atkinson et al., 2019) puntuazio gorenaren %2,56 soilik lor dezake. Bestalde, giza jokalari batentzako ez da izango batere erraza jokoaren amaierara heltzea.

Testuan oinarritutako jokoak erabiliz, errefortzu bidezko ikasketa ikertu daiteke ekintza espazio konbinatorial batean, lengoaia naturala balitz bezala. Hori dela eta, jokoek agenteei mundu erreal batean aritzen lagundu diezazkiete. Gainera, testuzko abentura jokoek munduaren obserbazio partzialarekin jardun behar duten RL algoritmoak hobetzea ahalbidetzen dute (Ammanabrolu and Riedl, 2019).

Testuan oinarritutako jokoetan jokalariak burutu nahi dituen ekintzak testu aginduak era-

biliz adierazten dira, hau da, lengoaiaren oinarrizko ulermen bat behar da atazak gainditzeko. Ulermen hori RL bidez, beste inongo gainbegiratze gabe, ikasi daitekeen ala ez irekita dagoen ikerketa galdera da. Ikerketa lerro horretan, RL bidez entrenatutako sistema batek munduari eta hizkuntzari buruz zenbat ikasi duen aztertzen da, eta bertan kokatzen da lan hau.

Proiektu hau garatzeko merkatuan eskuragarri dauden mundu birtualak aztertu egin dira, lengoaiaren aberastasuna kontuan hartuz. Horrez gain, garatutako agenteak ere ikusi egin dira, proiektuan zein ingurunetan aritzea erabakitzeko.

Behin hori aztertuta, TextWorld (Côté et al., 2018) ingurunean lan egitea erabaki da. TextWorld RL-ko agenteak entrenatzeko eta testatzeko sortutako ingurunea da. Horrez gain, joko berriak sortzeko edota beste ingurunetan sortutako jokoekin aritzeko aukera ematen du. Jokoak sortzeko prozesua parametrizatu ere egin daiteke, sortzen ari den munduan sortzailearen nahierara egin ahal izateko.

Proiektu honetan errefortzu bidezko ikasketako teknikak erabiliz testu-jokoetan aritzeko sistema bat ikertu da. Eredu horren funtzionamendua ulertzeko beharrezkoak diren oinarrizko kontzeptu teorikoak ere aztertu dira, aldaketaren bat egin nahi izanez gero, sistemaren xehetasunak menderatzea ezinbestekoa baita.

Laburbilduz, proiektu honen helburua sistema horrek TextWorld ingurunean errealitatea eta hizkuntzaren arteko zati konkretu bat RL bidez ikasi duen ala ez aztertzea da: etxe bateko gelen izen eta kokapenak. Horretarako, sistema joko desberdinetan entrenatuko da, horren ostean, beste jokoetan duen portaera ikusteko asmoz.

Memoria honetan proiektuaren nondik norakoak azaltzen dira, hainbat kapitulutan antolatuta. 2. kapituluan testu-jokoetan errefortzu bidezko ikasketako teknikak ulertzeko beharrezkoak diren oinarri teknikoak azaltzen dira. 3. kapituluan testu-jokoen nondik norakoak, haien historia, ingurune desberdinak eta joko motak azalduko dira. 4. kapituluan testu-jokoetan aritzeko sistema aurkeztuko da, A2C motako agente bat, hain zuzen ere. 5. kapituluan, aldiz, erabilitako sistema erabiliz lortutako emaitzak eta haien ebaluazioa egingo da. Azkenik, 6. kapituluan proiektuaren ondorioak eta etorkizunerako lana zein izan litzatekeen azalduko da.

2. KAPITULUA

Oinarri teknikoak

Atal honetan testu-jokoetarako errefortzu bidezko ikasketa ulertzeko beharrezkoak diren oinarri teknikoak lantzen dira.

2.1 Errefortzu bidezko ikasketa (RL)

Errefortzu bidezko ikasketan sistemari ez zaio esaten zein ekintza burutu behar duen, baizik eta sari maximoa lortzen duen ekintza segida zein den aurkitu behar du. Normalean, une batean burututako ekintza batek, sari bat eragiteaz gain, hurrengo egoera alda dezake. Hauek dira errefortzu bidezko ikasketaren bi ezaugarri nagusiak: proba-errore bilaketa eta saria berehala ez lortzearen posibilitatea. Azken hau dela eta, agenteak beharbada ez du jakingo burututako ekintza egokia izan den. Saria berehala lortzekotan, agenteak burututako ekintza egokia ala ez izan den jakingo luke. Agentea ekintza optimoak zeintzuk diren asmatzen saiatuko da, ingurunean ahalik eta hoberen mugitzeko.

Errefortzu bidezko ikasketaren historiak bi ildo nagusi ditu: proba-errore ikasketa eta *kontrol optimoa*ren arazoa programazio dinamikoa erabiliz. Bi ildoak independenteak ziren, hirugarren bat agertu arte, TD (*Temporal Difference*) Ikasketa, hain zuzen ere. Azken honen agerpenarekin hiru kontzeptuen artean erlazio bat sortu zen.

Kontrol optimoa kontzeptua 1950eko hamarkadaren amaiera aldera agertu zen, denboran zehar sistema dinamiko baten portaeraren neurri bat minimizatu edo maximizatzeko sistema bat sortzeko erronka definitzeko. Kontrol optimoen arazoei irtenbidea aurkitzeko,

Bellman (1958)-ek programazio dinamikoaren kontzeptua garatu zuen, *emaitza optimoaren ekuazioa* garatuz. Honez gain, Bellman (1957)-ek Markov-en Erabakitze-Prozesuak aurkeztu zituen, kontrol optimoaren bertsio estokastikoa, hain zuzen ere.

Hiru ildoak 1980ko hamarkadan konbinatu ziren, egun ezagutzen den errefortzu bidezko ikasketa sortzeko.

Ikasketa automatikoaren bi mota ezagunenak ikasketa gainbegiratuta eta ez-gainbegiratua dira. Dena den, RL badago ML klasearen barruan. Gainbegiratutako ikasketan etiketatutako lagin batetik ikasten da. Ikasketa mota honetan, sistemaren helburua patroi bat orokortzea da, sarrera eta espero den irteeraren artean, entrenamendu multzoan ez dauden kasuetan ondo aritu dadin. Ikasketa modu eraginkorra izan arren, ez da egokia interakziotik ikasteko.

Bestetik, ikasketa ez-gainbegiratua etiketatu gabeko datuen patroi bat aurkitzean datza, kasu honetan, irteera zein den ez da ezagutzen. Errefortzu bidezko ikasketa, ordea, beste ikasketa paradigma bat kontsideratzen da.

Errefortzu bidezko ikasketan datuak ez dira independenteak eta ez daude uniformeki banatuta. Agenteak denbora gehiago igaro dezake ingurunearen zonalde zehatz batean, eta beste zonalde batzuk ikusi barik gera daitezke, beti ere agenteari emandako denboraren arabera.

Errefortzu bidezko ikasketaren erronkarik handiena esplorazioa eta ustiapenaren arteko elkarrekintza da. RL-eko agente batek jadanik ezagunak dituen ekintzak burutzea nahiago izango du, ekintza berriak aukeratu baino. Aitzitik, ekintza horiek aurkitzeko aurretik burutu ez dituen ekintzak burutu behar izango ditu. Beraz, agenteak jadanik egin duena ustiatu behar du baina esploratu ere egin behar du etorkizunean ekintza hobeak lor ditzan. Horretarako, agentea pixkanaka-pixkanaka joan beharko da ekintzak aukeratzen eta burutzen, hobeak zeintzuk diren ikus dezan.

Errefortzu bidezko ikasketan ingurune batean lan egiten da, non egoerak, ekintzak eta sariak osagarri nagusiak diren. Ekintza bat burutu ostean, egoera aldatuko da eta agenteak sari bat lortuko du, positiboa ala negatiboa izan daitekeena.

Funtzionamendua hobeto ulertzeko adibide bat azaldu daiteke, Super Mario[1] jokoa erabiliz. Kasu honetan honako hauek izango lirateke aipatu beharreko elementuak:

- **Agentea**: Mario Bros.

[1] https://mario.nintendo.com/

2.1 irudia: Super Mario joko baten adibidea.

- **Egoera**: uneko jokoaren parada, esaterako, pantaila.
- **Ekintzak**: ezkerrera jo, eskumara jo eta salto egin.
- **Ingurunea**: joko bakoitzeko mundu birtuala.
- **Saria**: Jokoan zehar lortutako puntuak, Mario bizirik irauteak inplikatzen duena. Mario hiltzekotan, puntuak galduko dira.

Super Mario joko bat gainditzeko, azkeneko bandera bat lortu behar da. Joko osoan zehar, agentea sari positiboak lortzen egon da, baina Mario hiltzen bada momentuan sari positibo horiek galduko lirateke eta jokoa amaituko litzateke. Beraz, aurreko sariak ez dira nahiko, sari orokor bat beharrezkoa da. Hona hemen *espero den sari metatua*ren kontzeptua.

2.1.1 RL-eko elementuak

Errefortzu bidezko ikasketan ingurune batean lan egiten da, non egoerak, ekintzak eta sariak osagarri nagusiak diren. Ekintza bat burutu ostean, egoera aldatuko da eta agenteak sari bat lortuko du, positiboa ala negatiboa izan daitekeena. Egoera aldaketa bakoitzarekin agenteak ingurunearen behaketa berria jasoko du.

Aipatutako elementuez gain, beste lau elementu zerrenda daitezke: politika, sari-seinalea, balio funtzioa eta modeloa.

- **Politika**k (π) agentearen ikasketa portaera definitzen du. Esperientziatik ikasi daitekeen politika determinista izan daiteke, $a \in A$ ekintza multzoko (A) ekintza bat eta $s \in S$ egoera multzoko (S) egoera bat izanda,

$$a = \pi(s) \tag{2.1}$$

edo baita politika estokastikoa ere, non a_t eta s_t uneko ekintza eta egoera diren, hurrenez hurren,

$$\pi(a|s) = \mathbb{P}[a_t = a|s_t = s] \tag{2.2}$$

- **Sari-seinale**ak RL-eko arazo baten helburua definitzen du. Une bakoitzean, inguruneak agenteari **saria** deritzon zenbaki bat bidaltzen dio. Agentearen helburu bakarra sari osoa maximizatzea da.

- **Balio funtzio**ak ekintza bat egoera zehatz baten zein ona den adierazten du. Agenteari esango dio zer nolako saria espero behar duen ekintza bat egoera baten burutuz gero. Laburbilduz, etorkizunean espero diren sarien aurreikuspena da, egoeren kalitate maila neurtuz politikaren arabera, r_{t+i} egoeren sariak izanik,

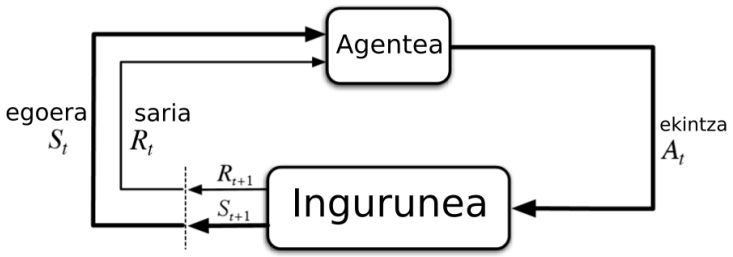
$$v_\pi(s_t) = \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i r_{t+i}|s_t = s], \forall s \in S \tag{2.3}$$

Bestalde, egoera batean ekintza zehatz bat burutzean espero den saria kalkulatzeko,

$$q_\pi(s,a) = \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i r_{t+i}|s_t = s, a_t = a] \tag{2.4}$$

q_π funtzioari π politikarentzako balio-ekintza funtzioa deritzo. Honako funtzio honek **Q-balioa** itzuliko du. Behin balio hori izanda, puntuazio altuena duen ekintza burutu daiteke. γ-ri deskontu faktorea deritzo, non $\gamma \in [0,1]$. Agenteari esango dio uneko sarietaz zenbat arduratu. $\gamma = 0$ bada, agenteak lehenengo sariaz bakarrik arduratuko da, $\gamma = 1$ bada, ordea, etorkizuneko sari guztietaz arduratuko da.

- **Modeloa**k inguruneak zer egingo duen aurreikusten du. Agentearen ingurumenaren errepresentazioa da, ingurumenak nola funtzionatzen duen agentearen ustea, hain zuzen ere. Modeloak erabiltzen dituzten RL-eko metodoei *model-based* deritze. Metodo hauek 2.1.4 azpiatalean deskribatzen dira. Modeloen trantsizio funtzioak



2.2 irudia: Agentearen eta ingurunearen arteko interakzio MEP batean. (Sutton and Barto, 2018)

(P) hurrengo egoera aurreikusiko du,

$$P = \mathbb{P}[s_{t+1} = s' | s_t = s, a_t = a] \tag{2.5}$$

s_{t+1} hurrengo egoeren probabilitatea kalkulatuko du, s_t uneko egoera eta agenteak burututako a_t ekintza kontuan hartuta. Bestetik, sari funtzioak (R), r_{t+1} berehalako saria aurreikusiko du, uneko egoera eta burututako ekintza kontuan hartuta,

$$R_s^a = \mathbb{E}[r_{t+1} | s_t = s, a_t = a] \tag{2.6}$$

2.1.2 Markov-en Erabakitze-Prozesuak

Markov-en Erabakitze-Prozesuak (**MEP**) erabakitze prozesu sekuentzialen formalizazioak dira, non ekintzek, berehalako sarietan eragiteaz gain, hurrengo egoeretan eta etorkizuneko sarietan ere eragiten duten (Sutton and Barto, 2018). MEPak errefortzu bidezko ikasketaren arazoen adierazpen matematikoak ere badira.

Une bakoitzean agentea egoera batetik bestera igaroko da, ekintza bat burutuz eta sari bat lortuz. Zehazki, agentearen eta ingurunearen elkarrekintza denbora tarte diskretu batean ematen da, $t = 0, 1, 2, 3...$ Denbora tarte bakoitzean (t) agenteak ingurunearen *egoera*ren errepresentazioren bat jasoko du, $s_t \in S$ eta egoera bakoitzean *ekintza* bat $a_t \in A(s_t)$ aukeratuko du. Hurrengo egoeran, s_{t+1} agenteak $r_{t+1} \in R$ saria jasoko du. Hortaz, esan daiteke ikasketa aro bakoitza, episodio ere deritzona, egoera, ekintza eta sarien sekuentzia bat bezala adierazi daitekeela. Egoera bakoitza aurreko egoeran eta ekintzaren araberakoa da eta agenteak ez daki zein izango den hurrengo egoera.

Prozesu oso honek Markov-en propietatea (Markov, 1954) betetzen du. Markov-en propietatearen arabera, prozesuaren etorkizuneko egoeren baldintzazko probabilitate-banaketa

egungo egoeraren araberakoa baino ez da, ez lehenago gertatutako gertaeren sekuentziaren araberakoa. Prozesu osoari Markov-en Erabakitze-Prozesua deitzen zaio.

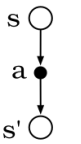
Bestalde, Partzialki Ikusgarriak diren Markov-en erabakitze-prozesuetan (PIMEP) ingurunea ez da guztiz ikusgarria, baizik eta agenteak unean bere aurrean dagoena ikusten du. Kasu honetan, agentea ingurunea eraikitzen joan behar da, momentura arte ikusi duena erabiliz eta, batzutan, probabilitatean oinarrituta dagoen hurbilketa bat eginez.

2.1.3 Oinarrizko kontzeptuak

TD Ikasketa (*Temporal Difference Learning*) errefortzu bidezko ikasketaren oinarrietako bat da. TD ikasketa gainbegiratu gabeko ikasketa teknika da, Monte Carlo metodoen eta Programazio Dinamikoaren arteko konbinaketa, hain zuzen ere, non agenteak aldagai baten balioa aurreikusten ikasten duen. Monte Carlo metodoetan $V(s_t)$ balioa episodioaren amaieran eguneratzen da, TD ikasketan, ordea, pausu bakoitzean eguneratzen da. TD metodorik sinpleenari $TD(0)$ edo *pausu bateko TD* deritzo,

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1} - V(s_t))] \tag{2.7}$$

non α konstante bat den (McClelland and Rumelhart, 1988).



TD(0)

2.3 irudia: TD(0)-ren adierazpen grafikoa, non s uneko egoera, a burututako ekintza eta s' hurrengo ekintza diren.

Q-ikasketa (Watkins and Dayan, 1992) aurretik aipatutako Q-balioan oinarritutako ikasketa da. Q-ikasketa algoritmoa politikarik gabeko algoritmoa da (1 algoritmoa),

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)) \tag{2.8}$$

ekuazioaz definituta. Kasu honetan, ikasitako ekintza-balio funtzioak, Q, zuzenean hurbiltzen du q_*, ekintza-balio funtzio optimoa, jarraitzen ari den politikari begiratu gabe.

Politikaren zeregina zein egoera-ekintza bikoteak bisitatzea izango da. Q-balioa matrize bat bezala adierazi daiteke, egoerak ilarak izanik eta ekintzak, aldiz, zutabeak.

Algoritmoa 1 Q-ikasketa (Sutton and Barto, 2018)

Algoritmoaren parametroa: pausu tamaina $\alpha \in (0,1]$, $\varepsilon > 0$
$Q(s,a), s \in S^+, a \in A(s)$ ausaz hasieratu, $Q(s_T,) = 0$ izan ezik
for episodio bakoitza $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T$ **do**
  for $t = 1..T-1$ **do**
    A ekintza S-tik aukeratu Q-ko politika erabiliz, ($\varepsilon-$greedy)
    A ekintza burutu, R saria eta S' egoera lortuz
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$
  end for
end for

Politika a ekintza t unean burutzearen probabilitatea da, s egoeran eta θ parametroekin. $\theta \in \mathbb{R}^{d'}$ notazioa erabiliko da politikaren parametroen bektorea adierazteko.

$$\pi(a|s,\theta) = Pr[a_t = a | s_t = s, \theta_t = \theta] \tag{2.9}$$

Politika edozein eratan parametriza daiteke, betiere $\pi(a|s,\theta)$ deribagarria bada bere parametroekiko, hau da, $\nabla\pi(a|s,\theta)$ existitzen bada $s \in S$, $a \in A(s)$ eta $\theta \in \mathbb{R}^{d'}$ guztietarako.

Policy gradient-en helburu-funtzioa honakoa da:

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} r_{t+1} | \pi_\theta] \tag{2.10}$$

ekuazioaz definitzen da. Beste era batera esanda, helburua politika bat ikastea da, zeinek espero den sari maximoa maximizatzen duena. Maximizazio problema denez, politika optimizatzen da *gradient ascent*-a erabiliz , θ parametroaren deribatu partziala erabiliz:

$$\theta \leftarrow \theta + \frac{\partial}{\partial\theta} J(\theta) \tag{2.11}$$

Policy Gradient teoremaren arabera, espero den sariaren deribatua sariaren produktuaren espero den balioa eta *policy*-aren logaritmoaren gradientea da. *Policy gradient*-a deribatzerako orduan honako ekuazioa lortzen da:

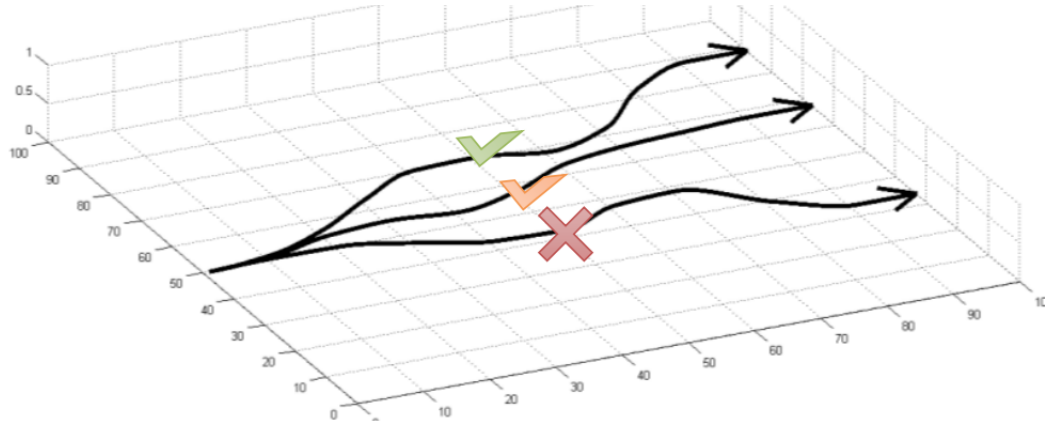
$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \tag{2.12}$$

REINFORCE algoritmoan, *Monte Carlo Policy Gradient* ere deitua, agenteak episodio baten τ ibilbidea hartuko du eta uneko politika erabiliz, politika parametroa eguneratuko du.

Algoritmoa 2 REINFORCE

θ ausaz hasieratu
for episodio bakoitza $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1..T-1$ **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
  end for
end for
return θ

REINFORCE algoritmoan, ikus 2 algoritmoa, politikaren parametroak Monte Carlo eguneraketen bidez eguneratzen da, hau da, ausazko balioak hartuz. Hori dela eta, logaritmoen probabilitateetan eta sari metatuen balioetan bariantza altua agertzen da, ibilbideak haien artean desbidera daitezkeelako, ikus 2.4 irudia.



2.4 irudia: *Policy gradient*-en balioen bariantza.

Beraz, bariantza altuak gradienteen errendimendua txikituko du eta ikasketa ez-egonkorra sortuko du. Gainera, ibilbide baten sari metatua 0 denean, zein ekintzak diren egokiak eta zeintzuk desegokiak ez da ikasiko, hori ikastea gradientearen helburua izanda ere.

Arazo hauek guztien ondorioz, *policy gradient* metodo sinpleak ezegonkorrak izan daitezke eta motelki konbergitu dezakete.

Arazoa saihesteko, hau da, bariantza txikitzeko eta egonkortasuna handitzeko, $b(s_t)$ *baseline* funtzio bat gehitu daiteke,

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))] \tag{2.13}$$

Metatutako saria txikituz, gradiente txikiagoak sortuko dira, beraz, eguneraketa gehiago baina txikiagoak egingo dira.

Hona hemen *baseline* funtzio batzuk, REINFORCE algoritmoaren aldaera nagusienak, *actor-critic* metodoak erabiltzen dutenak:

Q Actor-Critic Q^w balioa erabiltzen du bariantza txikitzeko

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^w(s,a)] \tag{2.14}$$

Advantage Actor-Critic A^w abantaila-balioa erabiltzen du gradientea eguneratzeko

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^w(s,a)] \tag{2.15}$$

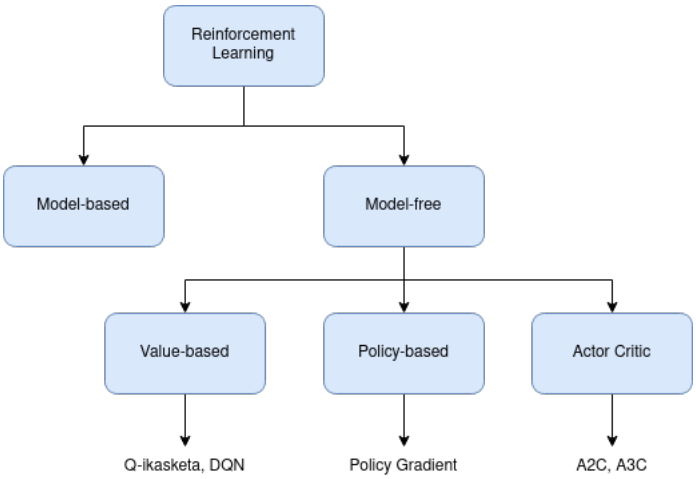
TD Actor-Critic TD errorean (δ) oinarritzen da eguneraketa egiteko

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \delta] \tag{2.16}$$

2.1.4 RL motak

Errefortzu bidezko ikasketako agenteak hainbat kategoriatan sailka daitezke, hona hemen garrantzitsuenetariko bi, 2.2 ataleko sistema ulertzeko oinarrizkoak direnak, ikus 2.5 irudia:

- Balioan oinarritutako (*value-based*) agenteek ekintza bakoitzaren kalitatea ebaluatuko dute uneko egoerearen arabera eta hoberena aukeratuko dute. *Q-learning* motako algoritmoak kategoria honetan aurki daitezke.
- Politikan oinarritutako (*policy-based*) metodoetan balio funtzioa agentearen barruan egon ordez, agentea balio funtzio optimoa aurkitzen saiatuko da. Hortaz, metodo hauek optimizazio problemak kontsidera daitezke, non balio maximoa lortzeko



2.5 irudia: RL metodoen sailkapena

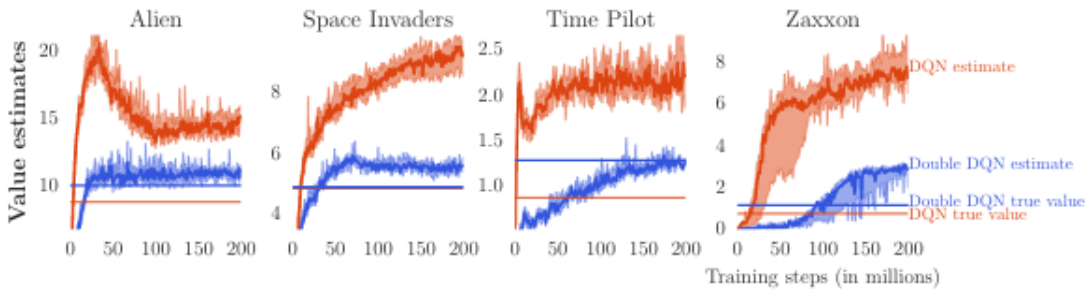
gradient ascent metodoa erabiltzen den. REINFORCE eta *policy gradient* algoritmoak kategoria honetan aurki daitezke.

- *Actor-Critic* metodoak aurreko bi metodoak konbinatzen lortzen dira, hau da, bai balioan eta bai politikan oinarritutako agenteak dira.
- *Model-Based* (modeloan oinarritutakoa) agenteak ingurunearen modelo bat eraikitzen saiatuko da. Honen ondoren, portaera optimizatzen saiatuko da.
- *Model-Free* (modelo barik) metodoetan agentea ez da saiatuko ingurunea ulertzen, baizik eta zuzenean balio funtzioan edota politikan arituko da.

2.1.5 Errefortzu bidezko ikasketa sakona (DRL)

Egoera multzo handia duten inguruneetan, errefortzu bidezko ikasketa tradizionala eraginkorra ez izatea gerta daiteke. Hori dela eta, errefortzu bidezko ikasketa sakonean neurona-sareak inplementatzen dira balio-funtzioak eta Q-balioaren funtzioak estimatzeko, errefortzu bidezko ikasketa sakona sortuz. Erabilitako sareak sinpleak izan ez ezik, konplexuagoak ere izan daitezke, konboluzionalak eta errekurrenteak ohikoenak izanik.

Deep Q Learning-ean (Mnih et al., 2015) neurona-sare bat erabiltzen da Q-balioaren funtzioa estimatzeko, DQN (*Deep Q-Network*) deritzona. Sarearen sarrera ingurunearen egoera izango da eta bere irteera ekintza bakoitzeko Q-balioa izango da. Behin ekintza guztien balioa izanda, balio maximoa duen ekintza aukeratuko du. Sarea entrenatzean, *TD*



2.6 irudia: Atari[2]inguruneko joko desberdinetako gain-estimazio joera (Hasselt et al., 2015)

Errorea erabiltzen da galera balioa kalkulatzeko, hurrengo egoerarentzako balio posible maximoa eta uneko Q-balioaren estimazioaren arteko diferentzia dena, Bellman (1958)-en planteatutako ekuazioaren arabera.

Q-Ikasketa sakonean *Experience Replay* kontzeptua agertzen da. Iraganeko ekintzak agenteak ahaztu ez ditzan, neurona sareari berriz pasatzen zaizkio iraganeko ekintzez osatutako multzo bat. Horri esker, agenteak iragana berriro bizi dezake eta horren ondorioz, bere memoria hobetu daiteke.

Dena den, DQN motako sareek arazo bat dute, *maximizatzeko joera* deritzona, ikus 2.6 irudia. Q-balioaren ekuazioa begiratuz,

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a')|s,a] \tag{2.17}$$

sareak ekintza batentzako Q-balioa gain-estimatzen badu, ekintza hori burutuko da eta hurrengo pausuan gain-estimatutako balio bera erabiliko da hurrengo estimazioa egiteko. Honen ondorioz, ezin da ebaluatu balio maximoa duen ekintza hoberena den ala ez.

Honen ondorioz, ikasketan maximizatzeko joera bat agertzen da, *Double Deep Q Network* motako sareek konpontzen dutena.

Double-DQN (Hasselt, 2010) bi DQN sareetan datza. Lehenengoaren zeregina DQN sinpleenaren berdina izango da, hau da, Q-balio maximoa duen ekintza aukeratuko du. Bigarrena, ordea, burututako ekintza horren ebaluazioaren arduraduna izango da. Bi sare independente erabiliz, joera gabeko sistema eraiki daiteke.

2.2 Actor-Critic algoritmoak

2.1.4 azpiatalean azaldutako Actor-Critic algoritmoak sakonago azalduko dira atal honetan. Horretarako, algoritmoaren definizioa eta bere mota desberdinak azalduko dira.

2.12 ekuazioa honako ekuazio honetan deskonposa daiteke:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)]\mathbb{E}_{r_{t+1},s_{t+1},\ldots,r_T,s_t}[G_t] \tag{2.18}$$

Bigarren zatia Q-balioaren definizioari dagokio, beraz, ekuazioa honako eran berridatzi daiteke:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q_w(s_t,a_t)] \tag{2.19}$$

Q-balioa Q-funtzioa parametrizatuz ikasi daiteke, neurona-sare bat erabiliz, goiko ekuazioan w notazioarekin adierazten dena. Ekuazio honi esker Actor-Critic metodoak garatu ziren, non:

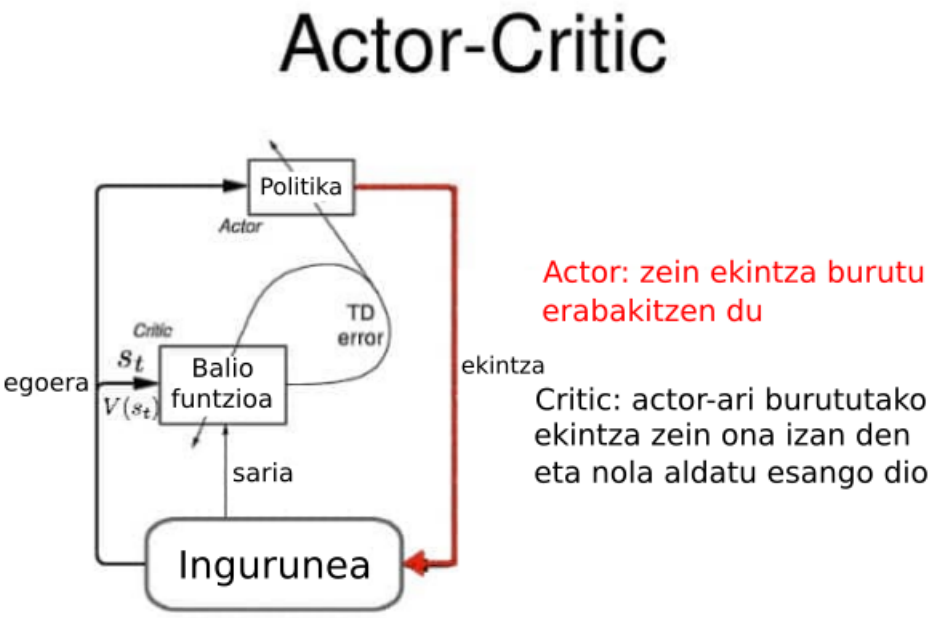
1. **Actor** zatia Critic zatiak esandakoaren arabera θ parametroak $\pi_\theta(a|s)$ politikan eguneratzeaz arduratuko da, *policy gradient*-etan bezala.

2. **Critic** zatia balio-funtzioaren w parametroaz eguneratzeaz arduratuko da, algoritmoaren arabera $V_w(s)$ egoera funtzioa edo $Q_w(a|s)$ balio-funtzioa izan daiteke.

2.7 irudian Actor-Critic egituraren errepresentazioa ikusi daiteke. Inguruneak sari bat emango du jasotako ekintzaren arabera. Saria, egoera berriarekin batera, balio funtzioa kalkulatzeko erabiliko da. Ondoren, politika erabiliko da hurrengo ekintza kalkulatzeko eta prozesua errepikatzeko, ikus 3 algoritmoa.

Actor-Critic metodorik sinpleenari *Q-Actor-Critic* deritzo, baina metodo konplexuagoak ere badaude, *Advantage Actor-Critic* (A2C) eta *Asynchronous Advantage Actor-Critic* (A3C).

[2] https://www.atari.com/



2.7 irudia: Actor-Critic eskema (Sutton and Barto, 2018)

Algoritmoa 3 Actor-Critic (Sarkar, 2018)

s, θ, w parametroak eta α_θ, α_w ikasketa ratioak hasieratu
while amaierako egoera ez izan **do**
  a ekintza aukeratu $a \sim \pi_\theta(a|s)$ eta burutu
  Hurrengo saria $r_t \sim R(s,a)$ eta hurrengo egoera $s' \sim P(s'|s,a)$ jaso
  $\delta \leftarrow r_t + \gamma v(s', w) - v(s, w)$ TD errorea
  $w \leftarrow w + \alpha_w \delta \nabla_w v(s, w)$ Critic balio-funtzioaren parametroen gradientearen eguneraketa
  $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \log \pi_\theta(a|s)$ Actor politikaren gradientearen parametroen eguneraketa
end while

Kasu honetan, v balio-funtzioa generikoa da, baina Q-funtzioa edota A funtzioa izan daiteke.

2.2.1 Advantage Actor-Critic (A2C)

A2C algoritmo sinkronoa da, A3C algoritmoaren (2.2.2) bertsio determinista dena. A3C algoritmoan agente bakoitza parametro globalekin era independentean aritzen da. Hori dela eta, agenteak bertsio desberdineko politikekin ari daitezke eta ondorioz, eguneratzen den balioa optimoa ez izatea gerta liteke. Horretarako, A2C algoritmoan koordinatzaile bat agente guztiak amaitzen dutela ziurtatzeaz arduratuko da, eta behin hori ziurtatuta, parametroak eguneratuko ditu, hurrengo iterazioan paraleloan aritzen diren actor-ek politika bera izan dezaten.

V balio-funtzioa *baseline* funtzioa bezala hartuz, Q-balioaren eta V balioaren arteko kenketa egin daiteke. Beraz, kalkulu hau eginez batez-besteko ekintzekin alderatuta, ekintza zehatz bat burutuz gero zer nolako hobekuntza lor daiteken jakin daiteke. Balio honi *abantaila-balioa* deritzo. A2C algoritmoan Critic zatiak A balioak ikasiko ditu. Honi esker, aurretik aipatutako bariantza gutxitu daiteke.

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t) \tag{2.20}$$

Q-balioaren definizioa erabiliz, abantaila-balioa honako eran adieraz daiteke:

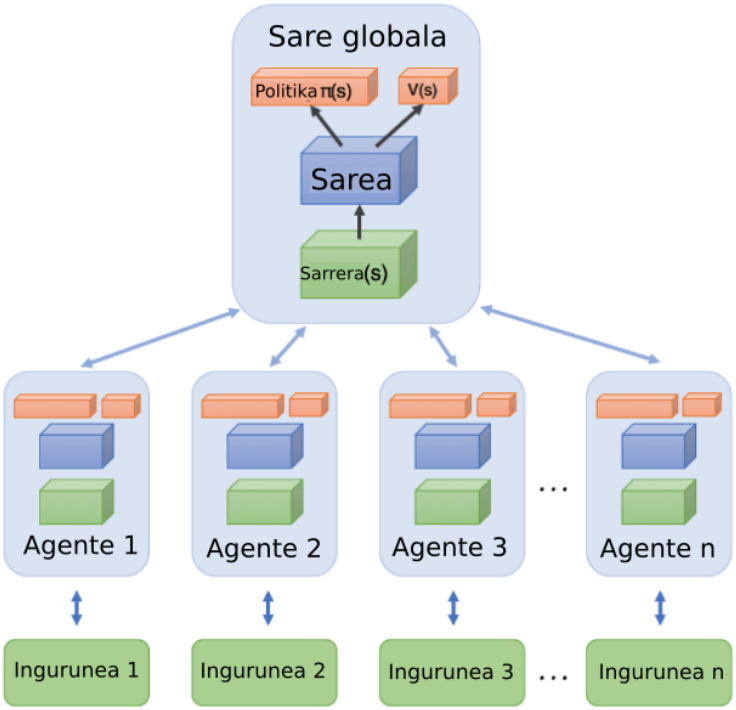
$$A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) \tag{2.21}$$

Beraz, helburu ekuazioa honela geratuko litzateke:

$$\nabla_\theta J(\theta) \sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_v(s_{t+1} - V_v(s_t)) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t) \tag{2.22}$$

2.2.2 Asynchronous Advantage Actor-Critic (A3C)

A3C algoritmoa (Mnih et al., 2016) entrenamendu paraleloan zentratutako *policy gradient* metodo bat da. A3C-n Critic zatiak balio-funtzioa ikasten du zenbait Actor paraleloan entrenatzen diren bitartean. Horretarako, hainbat ingurune paralelo sortzen dira, bakoitza bere ezaugarriekin.



2.8 irudia: A3C eskema (Sarkar, 2018)

Agenteak paraleloan entrenatzen dira eta periodikoki sare global bat eguneratzen da. Eguneraketak ez dira aldi berean burutzen, horregatik agente asinkronoa da. Eguneratze bakoitzaren ostean, agenteek sare globalaren parametroak atzituko dituzte eta haien esplorazio independentearekin jarraituko dute.

2.3 Neurona-sareak

Neurona-sareak giza burmuinean inspiratutako eredu konputazionalak dira. Neurona artifizialez osatutako grafo zuzendu bat bezala adierazi daitezke. Neuronak geruzetan antolatzen dira, era hierarkikoan, bi neurona geruza berean egongo dira baldin eta sarrera bera jasotzen badute.

Neurona bakoitzak balio bat kalkulatuko du, honako ekuazio hau erabiliz:

$$out = \phi(\sum_{i=0}^{n} w_i x_i) \tag{2.23}$$

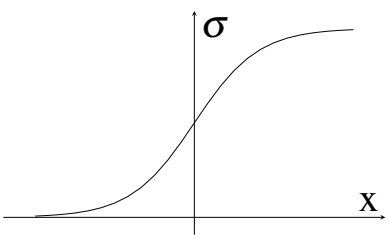
non ϕ aktibazio funtzioa, w neuronen pisuak eta x balioak diren,. Balio horrekin zer egin neurona sarearen arkitekturaren arabera erabakiko da; baliteke beste geruza baterako sarrera izatea edo irteerako balioa izatea.

2.3.1 Aktibazio-funtzioak

Aktibazio-funtzioak neurona baten irteera funtzioa definitzen du, beraz, esan daiteke, sarearen ahalmena kontrolatzen duen funtzioa dela. Hauek dira gehien erabiltzen diren aktibazio-funtzio batzuk:

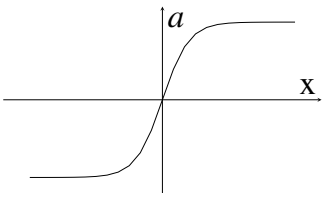
Funtzio sigmoidea

$$\sigma : \mathbb{R} \to (0,1), \qquad \sigma(x) = \frac{1}{1+e^{-x}}$$



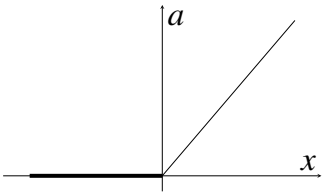
Tangente hiperbolikoa

$$\tanh : \mathbb{R} \to (-1,1), \qquad \tanh(x) = a(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



ReLU

$$ReLU : \mathbb{R} \to [0,\infty), \qquad ReLU(x) = a(x) = \max(0,x)$$



Softmax

$$softmax : \mathbb{R}^n \to [0,1]^n, \qquad softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Beste funtzioak ez bezala, *softmax* funtzioa ez zaio aplikatzen neurona bakar bati, baizik eta geruza bateko neurona guztiei. Oro har, *softmax* funtzioa balioak konparatzeko erabiltzen da, balio altuena edo baxuena zein neuronak duen ikusteko.

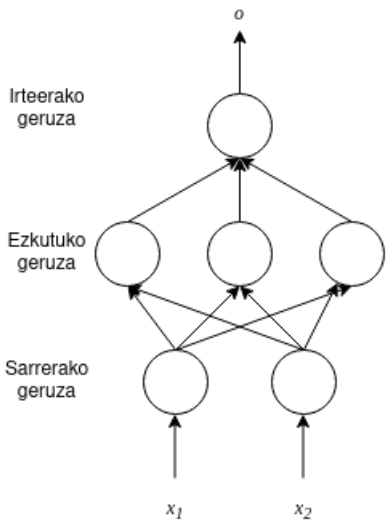
2.3.2 Arkitekturak

Neuronen arteko egituraren arabera, neurona-sareek arkitektura desberdinak izan ditzakete. Ikasketa atazaren arabera, arkitektura bat beste bat baino aproposagoa izan daiteke. Batzutan, arkitekturak konbinatzea eraginkorra izan daiteke ikasketa hobetzeko asmoz.

Multilayer perceptron

Arkitekturarik ezagunena eta sinpleenari *multilayer perceptron (MLP)* deritzo. Arkitektura honetan, neuronen arteko loturak sinpleak dira eta ez dago ziklorik. Neuronak geruza

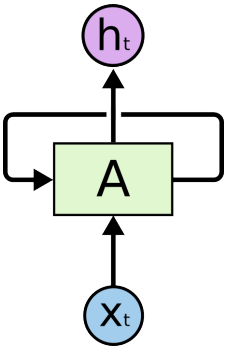
desberdinetan antolatzen dira, hiru motetan sailkatzen direnak: sarrera geruza, ezkutuko geruza eta irteera geruza. Ezkutuko geruza anitz egon daitezke, baina sarrera eta irteera geruza bakarra dago. Neurona sare mota honetan, geruza bateko neuronek aurreko geruzako neurona guztiekin daukate lotura.



2.9 irudia: *Multilayer Perceptron*

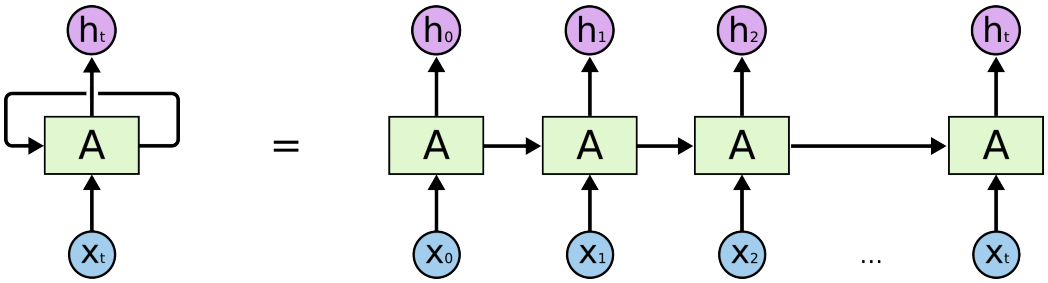
Sare errekurrentea

Sare errekurrenteetan neurona batek bere buruarekin izan ditzake konexioak. Honi esker, sareak memoria lortzeko aukera du, testuak prozesatzeko oso aproposa dena.



2.10 irudia: Sare errekurrentea
Iturria: https://colah.github.io/

2.10 irudian sare errekurrente bat azaltzen da, tolestuta. Sare mota honen egitura hobeto ulertzeko sarea destolestu daiteke, funtzionamendua hobeto ikus dadin (2.11 irudia).



2.11 irudia: Sare errekurrente destolestua
Iturria: https://colah.github.io/

Sare errekurrente sinpleetan $h_t \in \mathbb{R}^d$ uneko egoera ezkutua eguneratzeko $x_t \in \mathbb{R}^n$ uneko sarrera eta h_{t-1} aurreko ezkutuko egoeraren transformazio linealen ($W_x \in \mathbb{R}^{dxn}, W_h \in \mathbb{R}^{dxd}$) baturari, g funtzio ez-lineal bat aplikatzen zaio.

$$h_t = g(W_h \cdot h_{t-1} + W_x \cdot x_t)$$

Sare errekurrenteen pisuak eguneratzeko *backpropagation through time* (Werbos, 1990) algoritmoa erabiltzen da. Algoritmo hori *backpropagation* (Rumelhart et al., 1986) algoritmoaren aldaera bat da, RNNak entrenatzeko bereziki diseinatua.

Sare errekurrente batean une bakoitzean irteera bat dago, beraz, kostu-funtzioa irteera guztien gaineko errorearen batura bezala definitzen da:

$$J(\theta) = \sum_{t=1}^{T} J_t(\theta) \tag{2.24}$$

Sare errekurrenteeko gradientea kalkulatzerako orduan, katearen erregela aplikatzerako orduan arazo bat azaltzen da, desagertzen doan gradientea (*vanishing gradient problem*) deritzona.

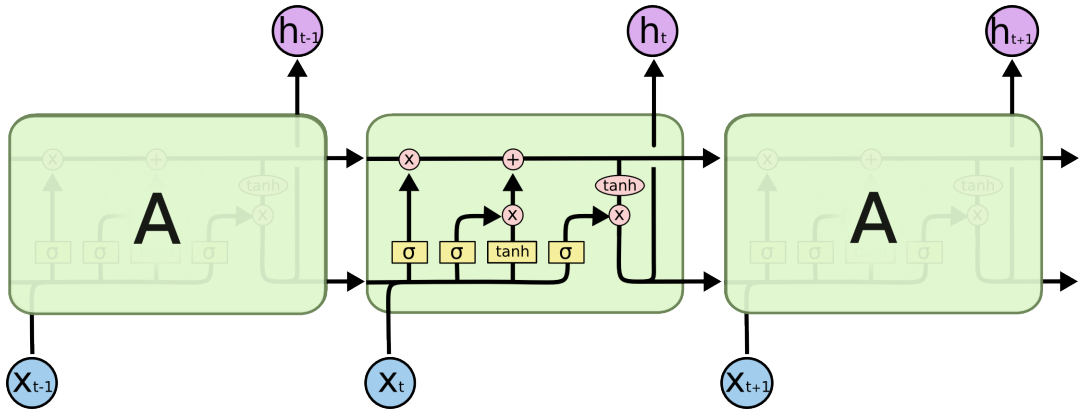
Hiru neuronako sare bateko azken neuronako gradientea kalkulatzeko, katearen erregela aplikatuz:

$$\frac{\partial J_3(\theta)}{\partial W_h} = \frac{\partial J_3(\theta)}{\partial h_3} \cdot \frac{\partial h_3}{\partial W_h} \tag{2.25}$$

Baina $\frac{\partial h_3}{\partial W_h}$ kalkulatzeko, aurreko ezkutuko egoeren balioak beharrezkoak dira:

$$h_3 = g(W_x \cdot x_3 + W_h \cdot h_2)$$
$$h_2 = g(W_x \cdot x_2 + W_h \cdot h_1)$$
$$h_1 = g(W_x \cdot x_1 + W_h \cdot h_0)$$
$$h_3 = g(W_x \cdot x_3 + W_h \cdot (g(W_x \cdot x_2 + W_h \cdot (g(W_x \cdot x_1 + W_h \cdot h_0)))))$$

Beraz, ekuazio bakoitzak bere gradienteak izango ditu eta katearen erregela aplikatuz, biderkadurak kalkulatzen joango dira, izandako gradientea txikituz edo, batzutan, handituz (honi *exploiding gradient* deritzo). Honen ondorioz, neurona sarearen pisuak ez-egokiak izatera hel daitezke.



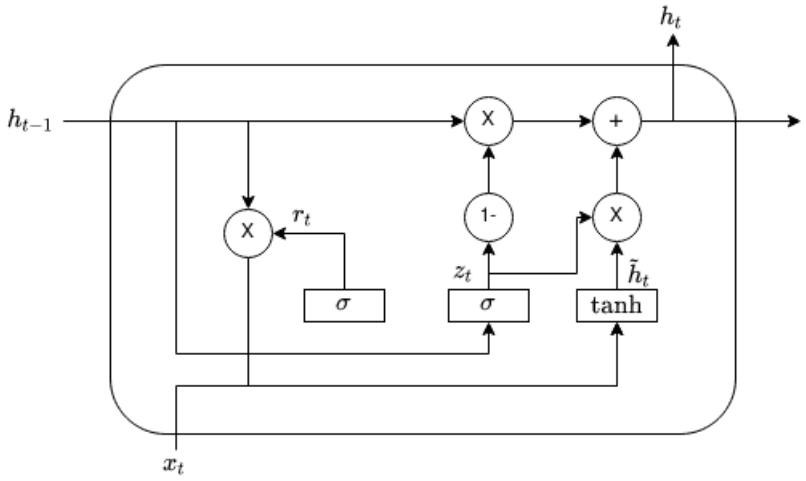
2.12 irudia: LSTM neurona
Iturria: https://colah.github.io/

LSTM (*Long Short-Term Memory*) motako neurona-sareak (Hochreiter and Schmidhuber, 1997) sare errekurrenteen *vanishing/exploiding gradient* arazoa konpondu nahian sortu ziren.

Horretarako, arkitektura berezi bat diseinatu zuten, hainbat atez konposatuta, ikusi 2.12 irudia. LSTM arkitekturan, neurona bakoitzak zenbait ate ditu sarearen informazioarekin zer egingo den erabakitzeko, hau da, informazioa ahaztuko den ala kontserbatuko den erabakitzeko.

LSTM neuronetan sartutako elementu berriak zelula-egoera (c_t) eta ahazteko atea (f_t) dira. c_t zelula-egoeran zer informazio mantenduko den erabakitzeko, h_{t-1} aurreko egoera ezkutua eta x_t uneko sarrera erabiliz f_t ahazteko atea kalkulatu behar da.

GRU (*Gated Recurrent Unit*) (Cho et al., 2014) LSTMaren aldaera sinplifikatua da. Ahazteko eta sarrera atea konbinatzen ditu eguneratze-atean eta zelula-egoera eta egoera ezkutua konbinatzen ditu (2.13 irudia).



2.13 irudia: GRU neurona

Ondorengo ekuazioen bidez modela daiteke GRU motako zelda bat:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

2.3.3 Atentzio-mekanismoa

Testu sekuentzia luzeak kodetzerako orduan zaila da informazio guztia gordetzea. Horretarako, atentzio-mekanismoak testu sekuentziako informazio garrantzitsua identifikatuko du.

Atentzio-mekanismoa (Bahdanau et al., 2014) lanean proposatu zen. Atentzioaren oinarrizko kontzeptua esaldi bakoitzeko bektore bakarra ikasi ordez, sarrerako sekuentziaren bektore espezifikoei arreta jartzea da.

Atentzioari esker, modeloak hitzei zein mailatan arreta jarri jakin dezake. Horretarako, irteerako hitz bakoitzeko testuinguru-bektore bat sortzen da, c_t. Ondoren, h_t uneko ezkutuko egoera bakoitza hartu eta puntuazio bat ematen zaio eta *softmax* funtzioa erabiliz

probabilitate bat esleitzen zaio hitz bakoitzari. Azkenik, probabilitateak ezkutuko egoeren pisudun batura kalkulatzeko erabiltzen dira.

Luong et al. (2015)-ek atentzio mekanismo sinpleago bat garatu zuten, bi aldaerekin: *Global* eta *Local*. Lehenengoak sarrerako elementu guztietan aplikatzen du atentzioa eta bigarrenak, ordea, elementu gutxi batzuetan.

2.4 Testuak kodetzen

Neurona-sareak testua prozesatu ahal izateko, bektore edo zenbaki eran adierazi behar da. Testu sekuentzia bat jasotzean, testua tokenetan banatzen da. Horretarako, zenbait metodo desberdin daude, esaterako, hutsunez hitzak banatuz edo karaktere bereziak erabiliz hitzak banatzeko.

Indexatutako hiztegian (*word2id*) *corpus*ean dagoen token bakoitzari indize bat esleitzen zaio. Hiztegia sortzen denean, ohikoa da hiztegiaren tamaina mugatzea, agerpen kopuruaren arabera. Tokenek *corpus*ean dituzten agerpen kopuruaren arabera ordenatzen dira eta agerpen kopuru altuenekoak soilik sartzen dira. Behin hori eginda, bakoitzari indize bat egokitzen zaio eta token bakoitza bere indizeagatik ordezkatzen da.

Esanahi-bektoreetan (*embedding*) tokenak dimentsio finituko bektore bidez errepresentatzen dira. Bektorea ausazko balioekin hasieratzen da eta entrenatu ahala balioak finduz doaz, hitzen arteko antzekotasuna agertuz. Dena den, baliteke aurre entrenatutako esanahi-bektoreak erabiltzea.

Aurre entrenatutako zenbait *embedding* eskuragai daude entrenamenduko zati hori ez egitea ahalbidetzen dutenak. Hauei esker, emaitza hobeak lor daitezke, izan ere, erabiltzen dituzten *corpus*ek izugarrizko tamaina dute, zenbat eta datu gehiago gero eta emaitza hobeak lortzen dira. Ezagunenak *word2vec* (Mikolov et al., 2013), GloVe (Pennington et al., 2014), ELMo (Peters et al., 2018) eta BERT (Devlin et al., 2018) dira.

2.5 Munduaren errepresentazioa

TextWorld-eko (Côté et al., 2018) ingurunea errepresentatzeko hainbat ezaugarri kodetu behar dira (3.2.1 azpiatala). Kodeketa era inplizituan edo era esplizituan egin daiteke.

2.5.1 Errepresentazio inplizitua

Errepresentazio inplizituan ez dago inolako modu egituraturik munduaren egitura kodetzeko. Oro har, ohiko neurona sareek errepresentazio inplizituarekin lan egiten dute eta ondorioz, erabiltzaileak ez du inolako erarik sistemak ikasi duena ikusteko. Errepresentazio inplizituan munduaren egitura testu obserbazioak erabiliz ikasten da, ondoren, neurona-sareen pisuak erabiliz eguneratzen direnak.

LSTM-DQN sistemak (Narasimhan et al., 2015) LSTM motako neuronak erabiltzen ditu o_t agentearen obserbazioa kodetzeko eta munduaren errepresentazioa egiteko. Objektuen eta ekintzen Q-balioa banaka kalkulatzen ditu eta multzo bakoitzeko balio altueneko elementua hartzen du burutuko den ekintza sortzeko, zeinen Q-balioa aurreko bien artekoen batez bestekoa izango da.

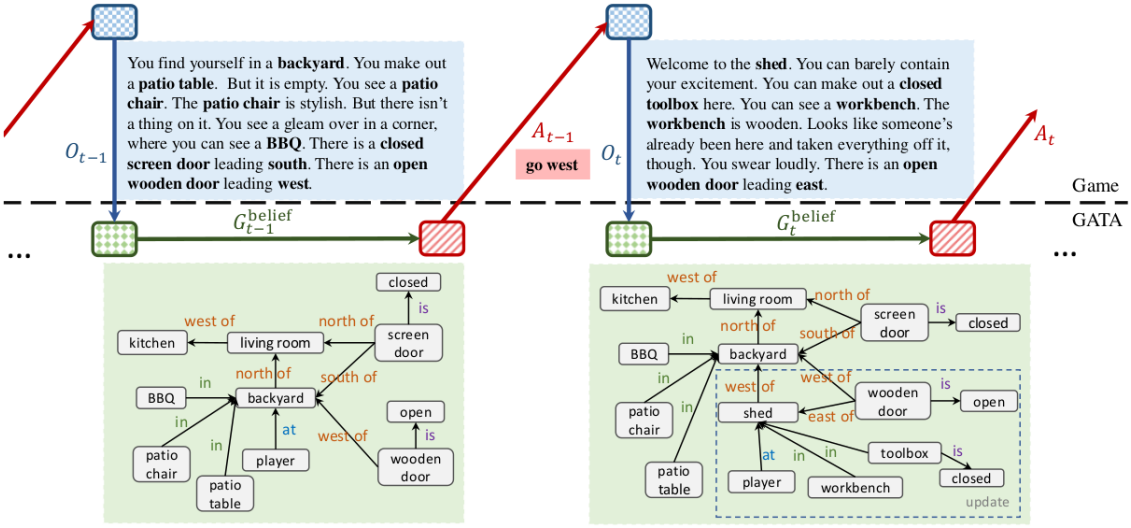
DRNN-ek (*Deep Reinforcement Relevance Network*) (He et al., 2016) bi neurona sare erabiltzen ditu, bat jokoaren egoera kodetzeko eta bestea, ekintza posibleak kodetzeko. Horren ostean, bien arteko konbinaketaren Q-balioak estimatzen ditu eta balio horien artean maximoa aukeratzen du.

2.5.2 Errepresentazio esplizitua

Bestetik, era esplizituan munduaren errepresentazio egituratua egiten da. Ezagutza-grafoa (KG) (*knowledge graph*) mundua errepresentatzeko erabili ohi den egitura da. Honetan, gelak, haien arteko konexioak, objektuak, jokalariaren objektuak eta bestelako entitateak gorde daitezke, nodoen eta ertzen bidez.

GATA (*Graph Aided Transformer Agent*) (Adhikari et al., 2020) izeneko sistemak (2.14 irudia) mundua era esplizituan errepresentatzen du, grafo zuzendu baten bitartez. Ingurunean aldaketa bat dagoen bakoitzean, grafoa eguneratzen da, entitate berriak grafoan gehituz edota aurretik zeudenen ezaugarriak aldatuz (posizio aldaketak, egoera aldaketak). Sistema honen berezitasuna hiru grafoen erabilera da, G^{seen}, G^{belief} eta G^{full}. Lehenengoa jokoa aurrera egin ahala eguneratzen doa, agenteak unera arte ikusi duenarekin; bigarrenak, ordea, agenteak munduan zer dagoen pentsatzen duena du eta azkenak, munduaren errepresentazio osoa du, TextWorld berak emandakoa.

KG-DQN (Ammanabrolu and Riedl, 2019) agenteak aipatutako KG erabiltzen du. Kasu honetan, ezagutza-grafoa Standford's OpenIE (Angeli et al., 2015) erabiliz eguneratzen da, beraz, ez dago inolako ikasketarik grafoa sortzerako garaian. Aitzitik, agenteak



2.14 irudia: GATA sistema (Adhikari et al., 2020) joko bat jokatzen bere G^{belief} eguneratuz. A_{t-1} ekintza burutu ostean, inguruneak O_t testuzko behaketa bueltatuko du. O_t behaketan eta G^{belief}_{t-1} grafoan oinarrituz, agenteak G^{belief}_t eguneratuko du eta A_t ekintza berria aukeratuko du. Argazkian, laukizko kutxa urdinak jokoaren funtzionamendua adierazten du, diamanteko kutxa berdea grafo eguneratzailea da eta marraz dagoen kutxa gorria ekintza hautatzailea da.

grafoan dauden egoera eta ekintza posibleen arabera DQN (2.1.5) teknika erabiltzen du hurrengo ekintza aukeratzeko.

Sistema honen bertsio berriago batek, KG-A2C (Ammanabrolu and Hausknecht, 2020), KG modu berean erabiltzen du, baina ekintza hautaketa egiterako orduan, A2C algoritmoa erabiltzen du, zeinek emaitza hobeak lortzen dituen.

3. KAPITULUA

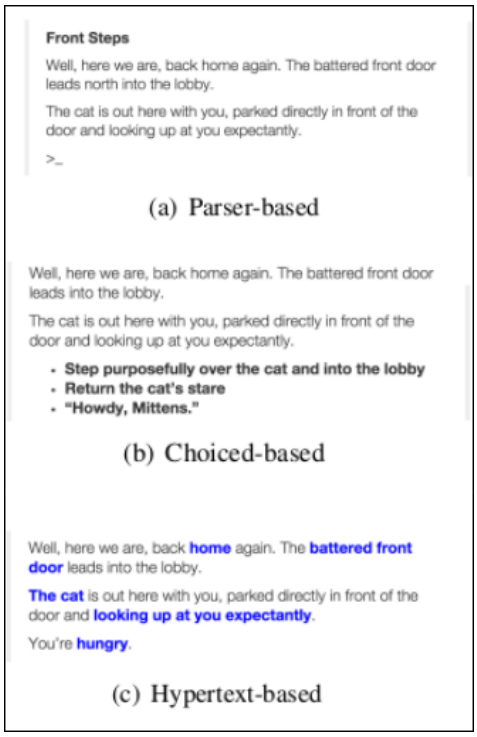
Testu-jokoak

Testu-jokoak simulazio konplexu eta interaktiboak dira, non jokoaren interakzio guztiak testu bidez egiten diren. Jokoan ager daitezkeen egoera desberdinen deskribapenak, jokalariak adierazten dituen ekintzak eta jokoan aurkitu daitezkeen objektuen deskribapenak testuz adierazten dira. Mota honetako jokoetan arrakasta izateko, lengoaia naturalaren ulermenaz gain, esploratze gaitasunak, memoria eta zentzuz jokatzea ezinbestekoak dira. Hori dela eta, ingurune ezin hobeak dira lengoaia naturalaren prozesamenduko (LNP) teknikak eta RL-ko agenteak garatzeko.

Testu abenturen inguruneak espazio konbinatorial eta konposizional bezala defini daitezke, hortaz, mundu errealarekin aldera daitezke. Honez gain, partzialki ikusgarriak diren eremuak dira, hau da, mundu osoa ezin da aldi berean ikusi, unean jokalaria dagoen lekua baizik.

Testuan oinarritutako jokoetan aritzeko, esan bezala, testua besterik ez da erabiltzen. Horrela, normalean komando lerroko terminala erabiliz jokatzen da. Jokoa abiarazten denean, helburua eta hasierako egoera azaltzen dira. Jokoaren ezarpenen arabera helburua luzeagoa ala motzagoa, xehetasun gehiagorekin ala gutxiagorekin edota errazagoa ala zailagoa izango da.

Testu abenturak hiru mota desberdinetan klasifika daitezke, jokalariak komandoak nola adieraz ditzakeen arabera (3.1 irudia): *parser-based*, *choice-based* (aukeretan oinarritutakoak) eta *hypertext-based* (hipertestuan oinarritutakoak). Lehenengoan, jokalariak nahi duen komandoa idatziko du eta jokoaren *parser*-a (interpretatzailea) komandoa ulertzeaz arduratuko da. Bigarrenean, ordea, jokoak hainbat aukera emango ditu eta jokalariak ho-



3.1 irudia: Joko motak. (He et al., 2016)

rietako bat aukeratu beharko du. Azkenik, *hypertext-based* motan jokoaren deskribapenean hitz batzuk loturak bezala agertuko dira eta jokalaria haietan klik egiteko gai izango da.

Testuan oinarritutako jokoak, *Interactive Fiction (IF)* deituak baita ere, 1980ko hamarkadan hasi ziren garrantzia hartzen, batez ere *Zork 1* jokoaren agerpenarekin. Garai hartan, ordenagailuen ezaugarri teknikoak zirela eta, gaur egun arruntagoak diren irudietan oinarritutako jokoek denbora asko behar zuten kargatzeko eta jokatzeko prest egoteko. Testuan oinarritutako jokoek, aitzitik, ez zuten hainbeste denbora behar eta ez zuten hainbeste baliabide kontsumitzen. Testuan oinarritzen direnez, hizkuntza ezaugarririk garrantzitsuena dela esan daiteke. Hori dela eta, badaude hizkuntza ezberdineko IF komunitateak, esaterako, ingelesa eta frantsesa. Hainbat kode irekiko plataformei esker (TADS[1], Alan[2]) jokoak sortzea beti egon da erabiltzailearen eskura. Aipatzekoa da Inform plataforma, 1993an sortua. Egun, azken bertsioa Inform 7[3] da, hain zuzen ere, TextWorld-ek jokoak konpilatzeko erabiltzen duen plataforma da.

Testu-jokoen egiturari dagokionez, joko bakoitzean mundu desberdin bat sortzen da. Mun-

[1] https://www.tads.org/
[2] https://www.alanif.se
[3] http://inform7.com/

duan hainbat gela daude, bakoitza zenbait objektuekin. Gelak haien artean lotuta egon daitezke, normalean korridoreen bitartez. Korridoreetan ateak egon daitezke eta batzutan beharrezkoa izango da ate horiek zabaltzea gela batetik bestera pasatzeko. Elementu guztiek mapa bat osatzen dute. Mapan zehar ibiltzeko jokalariak nabigazio komandoak erabil ditzake, esaterako, *go* hitza norabide batekin lotuta (*north, south*, etab.) Esan bezala, gela bakoitzean objektuak egon daitezke eta objektu bakoitzak atributuak izan ditzake, adibidez, *container* motatako objektu batek *open, closed* atributuak izan ditzake. Ezaugarri hauek 3.2.1 atalean zehaztuta daude.

3.1 Partzialki ikusgarriak diren Markov-en Erabakitze-Prozesuak

Côté et al. (2018)-ren arabera, testuan oinarritutako jokoak partzialki ikusgarriak diren Markov-en erabakitze-prozesuak kontsidera daitezke.

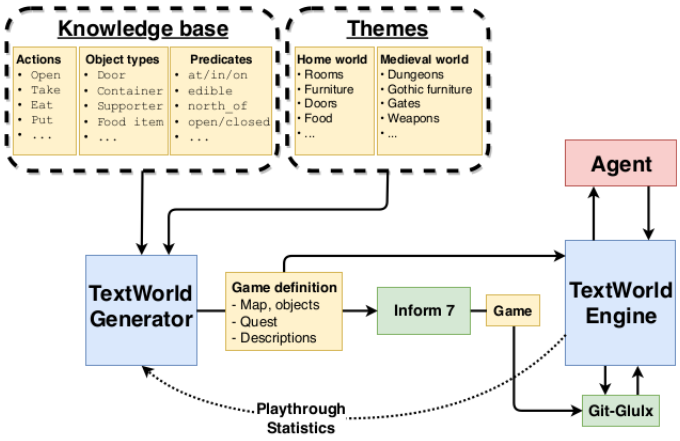
Beraz, joko bat prozesu bat bezala defini daiteke, zazpi elementuko tupla baten bidez: *(S,T, A, Ω, O, R, γ)* non S ingurunearen egoeren multzoa den, T egoeren arteko trantsizio posibleen probabilitateak dira, A testuzko komandoak eratzeko hitz posibleak dira, ω jokoaren behaketen multzoa da, O baldintzapeko behaketen probabilitateen multzoa da, $R : S\,x\,A \rightarrow \mathbb{R}$ sari-funtzioa da eta $\gamma \in [0,1]$ deskontu faktorea da.

Ingurunearen egoerak (S) joko osoaren informazioa du, hots, gelak, objektuak, jokalaria eta bere gauzak, etab. t uneko egoera s_t bezala adierazten da. Agenteak testu komandoa sortzen duenean, jokoak s_{t+1} egoerara pasatuko da, $T(s_{t+1}|s_t, c_t)$ trantsizio probabilitatearekin.

Ekintzak (A) t txanda bakoitzean agenteak testu komando bat sortuko du, c_t. *Parser-based* jokoen interpretatzaileak edozein karaktere kate onartuko du, baina jokoak komando zehatz batzuk baino ez ditu ulertuko.

Behaketak (Ω) Agenteak jasotzen duen testuari behaketa deritzo. t unean jasotzen den behaketa, $o_t \in \Omega$ ingurunearen egoeraren eta aurreko komandoaren araberakoak izango dira, izan ere, O izendun funtzioak $O(o_t|s_t, c_{t-1}$ probabilitatea kalkulatuko du eta erakutsi behar den informazioa aukeratuko du.

Sari-funtzioa (R) jokoak agenteari emango dion saria kalkulatzen du, $r_t = R(s_t, a_t)$. Agentearen helburua sarietatik espero den deskontatutako zenbatekoa maximizatzea da.



3.2 irudia: *Framework*-aren egitura. (Côté et al., 2018) Urdinez dauden osagaiak jokoaren sortzailea (*generator*) eta jokoaren motorra (*engine*. Berdez dauden osagaiak, ordea, *Inform 7* eta *Git-Glulx* TextWorld-ek beharrezkoak dituen liburutegiak dira. Gorriz dagoen osagaia agentea da. Agentea bai jokalaria edo bai agente neuronala izan daiteke, baina erabiltzailearen ardura izango da inguruneari agenteaz hornitzea. Azkenik, horiz dauden osagaiak ingurune beraren osagaiak dira: ezagutza-basea, jokoaren definizioa (mapa, objektuak, etab.) eta joko bera.

3.2 TextWorld ikasketa-ingurunea

TextWorld *reinforcement learning* (RL)-ean oinarritutako agenteak testu-jokoetan entrenatzeko eta testatzeko ingurunea da. Jokoak objektu kopuru, helburu, zailtasun eta bestelako parametroen arabera sortzeko aukera ere ematen du. Bertako ingurunean sortutako jokoetara eta beste testuan oinarritutako jokoetara (adibidez, Zork I (Blank and Lebling, 1980)) jokatzeko aukera ematen du. 3.2 irudian TextWorld *framework*-aren arkitektura azaltzen da.

3.2.1 TextWorld inguruneko elementuak

TextWorld-en zenbait elementu aurki daitezke, hona hemen haien zerrenda bat:

- Jokalaria (*Player*): *P* hizkiaz eta pertsona baten irudiaz errepresentatzen da.
- Janaria (*Food*): sagar baten irudiaz errepresentatzen da. Mota honetako objektuek atributu berezi bat izan dezakete: *edible* (jangarria). Horrela izanez gero, jokalariak jateko aukera izango du.

- Edukiontzia (*Container*): kutxa baten irudiaz errepresentatzen da. Aingura baten irudia ondoan edukiz gero, objektua gelan finkatuta egongo da, hau da, jokalariak ezin izango du hartu. Azkenik, mota honetako objektuek zabalik edo itxita (*open, closed, locked, unlocked*) egon daitezke, *locked* kasuan giltza bat beharko da zabaltzeko.
- Euskarria (*Supporter*): mahai baten irudiaz errepresentatzen da. Edukiontziak bezala, aingura batekin agertu ahal dira. Mota honetako objektuak beste motatakoak gainean izateko erabiltzen dira.
- Atea (*Door*): ate baten irudiaz errepresentatzen dira. Edukiontzien atributu berdinak dituzte eta haiek bezala, batzutan giltza baten beharra dago zabaltzeko.
- Giltza (*Key*): giltza baten irudiaz errepresentatzen da. Edukiontziak edota ateak zabaltzeko balio du. Bere izena eta zabaltzen duen elementuarena ezaugarri bera dute, adibidez, *red key* (giltza gorria) eta *red chest* (kutxa gorria).
- Korridorea (*Corridor*): Gelak lotzeko balio dute. Bi gelen arteko lerroek errepresentatzen dute. Ateak hor ipintzen dira.
- Objektua (*Object*): mota honetako objektuak gauza arruntak dira, esaterako, pala, xaboia, etab. Hodei baten irudiaz errepresentatzen dira.

3.2.2 Joko motak

TextWorld-ek hiru joko mota eskaintzen ditu: *simple* (sinplea), *treasure hunter* (altxorraren bila) eta *coin collector* (txanponak batu). Mota bakoitzaren adibide bat B eranskinean eskuragai dago.

- *Simple*: Joko mota hau ohikoena da. 3.2.4 azpiatalean azaltzen den jokoa mota honetakoa da. Agente gehienak erabiltzen duten joko mota da, izan ere, jokoak nahierara egin daitezke edota Python-eko *framework* berak egin ditzake. Jokoen sormenaren xehetasunak 3.2.3 azpiatalean daude azalduta.
- *Treasure hunter*: sortutako munduan bi objektu ipini eta agenteak horietako bat lortu behar du, hasierako mezuan esango zaiona, hain zuzen ere. Horretarako, beharrezkoa izan daiteke giltzak edo beste motatako objektuak lortzea. Erronka mota hau Parisotto and Salakhutdinov (2017)-k proposatutakoaren 3D bertsioa da. Hiru zailtasun maila desberdintzen dira:

  - **Maila: 1-10** (erreza): bost gela daude, zeintzuk hutsik dauden bi objektuak dauden gelak izan ezik. Geletatik aritzeko ez dago aterik. Helburuaren luzera batetik bostera linealki handituz doa.
  - **Maila: 11-20** (ertaina): hamar gela daude, edukiontziak izan ditzaketenak. Lortu behar den objektua horietako edukiontziren baten egon daiteke. Geletatik aritzeko ateak daude. Helburuaren luzera bitik hamarrera linealki handituz doa.
  - **Maila: 21-30** (zaila): hogei gela daude, non edukiontziak egon daitezken. Edukiontzi horiek zabaltzeko giltzak beharrezkoak dira. Geletatik aritzeko ate itxiak daude, zabaltzeko giltzen beharra dutenak. Helburuaren luzera hirutik hogeira linealki handitzen doa.

- *Coin collector*: Joko mota honetan munduak gela kopuru zehatz bat du, aukeratutako mailaren arabera. Agentea maparen mutur batean hasten da eta jaso beharreko txanpona beste muturrean egongo da. Munduan ez dago lortu behar den txanpona ez den beste objekturik. Hiru zailtasun maila desberdintzen dira:
  - **Maila: 1-100**: gela kopurua eta helburuaren luzera mailaren zenbakia bera dira.
  - **Maila: 101-200**: gela kopurua = 2 * (maila * % 100) eta helburuaren luzera = maila % 100.
  - **Maila: 201-300**: gela kopurua = 3 * (maila % 100) eta helburuaren luzera = maila % 100.
- *Custom* (Neurrira egindakoa): Joko mota hau sortzeko zailena da, hainbat parametro zehaztu behar baitira mundua sortzeko.

3.2.3 Jokoak sortzen

Jokoen sorkuntza hiru zatitan desberdindu daiteke: munduaren sorkuntza, testuaren sorkuntza eta helburuaren sorkuntza:

- **Munduaren sorkuntza**. TextWorld-ek jokoak ausaz sortzeko aukera badu, *Random Walk* (Pearson, 1905) algoritmoa erabiliz. Mapa sortzerako orduan, zenbait parametro zehaztu daiteke: gela kopurua, objektu kopurua, etab. Behin mapa sortuta, objektuak geletan zehar kokatuko dira.

- **Helburuaren sorkuntza**. Helburuari buruz hitz egiteko, sarri *quest* hitza erabiltzen da, baina irabazteko jarraitu beharreko ekintza guztiak ere izan daiteke. Bide zikliKoak ekiditeko asmoz, ekintzetan mendetasun mugak ezartzen dira. Hori dela eta, a_t ekintza a_{t-1} ekintzaren araberakoa izango da baldin eta soilik a_{t-1} ekintzaren eskumako zatiak a_t ekintzak behar dituen ezker aldeko baliabideak sortzen baditu. Logikako erregelak 3.2.5 azpiatalean azaltzen dira.

- **Testuaren sorkuntza**. Jokoaren objektuen izenak, gelen deskripzioak eta beste elementuen izenak sortzeko testuinguru gabeko gramatika (TGG) (Chomsky, 1956) erabiltzen da. Testua txantiloiak erabiliz sortzen da. Txantiloiek objektuen izenekin bete beharreko hutsuneak dituzte eta baliteke objektu batzuk izenondoekin batera joatea. Hona hemen testuaren sorkuntzaren deskribapen zehatzagoa, elementuz sailkatuta:

  - **Objektuen izenak** TGG multzotik ausaz hartzen dira eta objektu bakoitzari egokitzen zaie. Horretarako, objektu mota zein den besterik ez da jakin behar. Objektu baten izena bitan deskonposa daiteke: izena eta adjektiboa, hautazkoa dena. Adjektiboak lagungarriak izan daitezke jokalariarentzat, esaterako, giltza baten adjektiboa eta zabaltzen duenaren adjektiboa berdinak izango dira.
  - **Gelaren deskribapena** gelan dauden objektuen deskribapenaren konkatenazioa da. Horrez gain, dauden irteerak ere aipatzen dira. Dinamikoki eguneratzen da objektuen egoera aldatzen denean.
  - **Helburuaren jarraibideek** ekintza guztiak deskriba ditzake, errazagoa dena, azken ekintza (zailagoa) edo jokalariari helburua asmarazi diezaioke.

  Azkenik, TextWorld bi joko gaien artean aukeratzeko aukera amaten du: *house* eta *basic*. Lehenengoak, lehenetsia, jokoa deskribatzen du etxe moderno bat izango balitz bezala. Bigarrenak, ordea, gramatika sinpleago bat erabiltzen du, adjektiborik gabe, esaterako, eta objektuak desberdintzeko zenbakiak gehitzen dira izenaren amaieran.

Jokoak sortzeko bi metodo eskaintzen ditu TextWorld *framework*-ak: terminala erabiliz edo *GameMaker* Python-eko paketearen klasea erabiliz.

1. Terminala. Jokoaren sorkuntza prozesua parametrizatu egin daiteke. Lehenik eta behin, joko mota zehaztu behar da. Mota bakoitzeko jokoak parametro propioak ditu, esaterako, *coin collector* eta *treasure hunter* jokoetan **maila** zehaztu behar

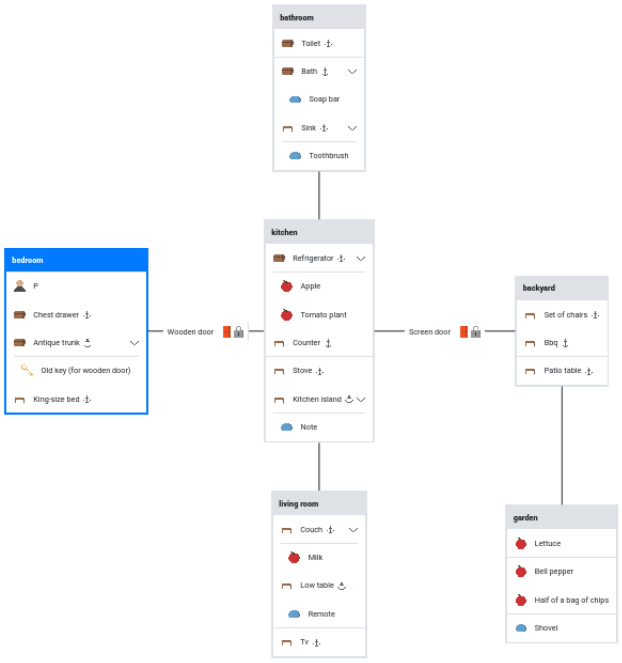
da, baina badaude parametro batzuk joko guztiek dituztenak, adibidez, jokoaren jarraibideak anbiguoak izatea aukera daiteke. Kasu horretan, jarraibideak jarraitzea zailagoa izango litzateke. Adibidez, *red chest* (kutxa gorria) *red container* (edukiontzi gorria) hitzak erabiliz definitu daiteke, *chest container* multzoan sartuta baitago. Horrelako anbiguotasunek jokoaren ulermena zaildu dezakete agente batentzat. *Simple* motari dagokionez, sarien eta helburuaren parametrizazioa nahitaezkoa da. Sarien konfigurazioari begira, hiru aukera desberdin daude: *dense, balanced, sparse* (sarri, orekatu, eskas). Lehenengoaren kasuan agenteak saria (puntu bat) jasoko du helburura hurbiltzen dion ekintza bat burutzean. Bigarren kasuan, agenteak noizbehinka lortuko ditu puntuak eta azkenean, ordea, helburua lortzean soilik lortuko da puntua. Helburuari dagokionez, beste hiru aukera daude eskuragai: *brief, detailed, none* (laburra, zehatza, bat ere ez). Laburrean agenteak zer egin behar duen azalduko da, inolako zehaztapenik gabe. Zehaztuan, ordea, helburua eta hura nola lortu azalduko da. Azkenik, *none* aukeran ez da helbururik azalduko eta agenteak asmatu behar izango du.

2. *GameMaker*. Lehenengo metodoan ez bezala, jokoak guztiz neurrira egitea ere posible da. Horretarako, sortzaileak mundu bat sortu beharko du 3.2.1 azpiatalean zehaztutako elementuak gehituz. Haien artean erlazioa duten elementuak zeintzuk diren ere zehaztu beharra dago, esaterako, giltza bat gehitzen bada, giltza horrek zer irekitzen duen zehaztu behar da. Behin elementuak gehituta, jokalaria gela baten jarri behar da. Jokoaren zailtasun maila handitu nahi bada, ausazko objektuak gehitzeko aukera ere badago. Munduaren egitura guztiz amaituta dagoenean, jokoaren helburua zehazten da. Horretarako, sortzaileak eman beharreko pausuak zehaztu behar ditu, sortu berri izan den munduan beharrezkoak diren pausoak burutuz azkeneko helburua erdietsi arte. 3.3 irudiko jokoa nola sortu C eranskinean dago.

Joko bat sortzerako orduan, oso garrantzitsua da helburua lor daitekeela ziurtatzea, bestela, ez da baliozkoa izango.

3.2.4 Joko sinple baten adibidea

TextWorld ingurunea hobeto ulertzeko, 3.3 irudiko jokoaren azalpena egingo da. Irudiak jokoaren hasierako egoera irudikatzen du. Hasteko, lauki bat urdinez ikus daiteke, jokalaria dagoen gela. Jokalaria mugitzen bada, laukiaren kolorea grisa izango da eta gela berriaren laukia kolorez aldatuko da. Gainera, gelako objektuak ikus daitezke: bi *contai-*



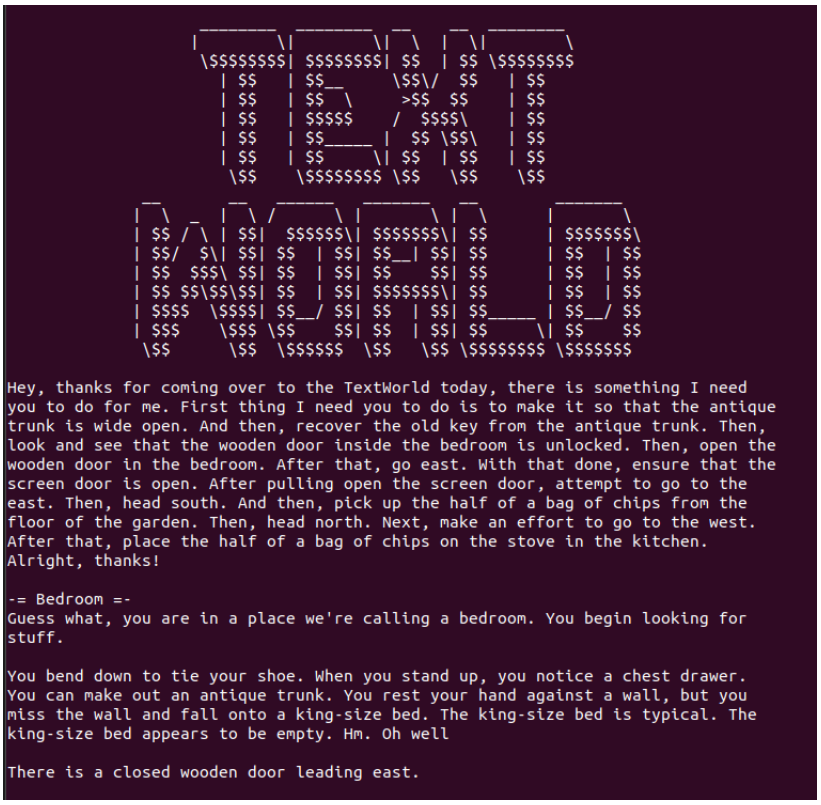
3.3 irudia: Joko sinplea.

ner motatako objektuak (*Chest drawer, Antique trunk*), *key* motako objektu bat (*Old key (for wooden door)* eta *supporter* motako beste objektu bat (*King-size bed*). Ikusi daitekeenez, *container* eta *supporter* motatako objektuek aingura bat dute haien ondoan. Sinbolo hori agertzen den bakoitzean, objektua finkatuta dago, beraz, jokalariak ezin du bere posizioa aldatu ezta bere inbentarioan sartu. Irudian, inbentarioa hutsik dago, normalean jokoa hasten denean inbentarioa hutsik dago, baina baliteke inbentarioan objekturen bat egotea ere. Gelatik irteteko irteera bakarra dago (*wooden door*), giltzarrapo itxi batekin. Horren arabera, giltza bat beharko da, aurretik aipatutakoa, izan ere. Gainerako geletan beste objektu motak begietsi daitezke (*Apple, Note*, etab.). Horrez gain, badaude gelak haien artean lotuta daudenak korridore sinpleak erabiliz, aterik gabe.

3.4 irudian joko honen hasierako pantaila agertzen da.

3.2.5 TextWorld eta Markov-en Erabakitze-Prozesuak

TextWorld Markov-en erabakitze-prozesu (MEP) bat bezala defini daiteke, bost elementuko hurrengo tuplaz: (S, A, T, R, γ) (Côté et al., 2018), non S ingurunearen egoeren multzoa den, A ekintzen multzoa da, $T(s_t, a, s_{t+1}) = P(s_{t+1}|s_t, a)$ egoera trantsizio funtzioa



3.4 irudia: Azaldutako jokoaren *quest*-a.

da, uneko egoeraren s_t eta erabakitako ekintzaren a araberakoa dena, $R : SxA \rightarrow \mathbb{R}$ sarifuntzioa da eta $\gamma \in [0,1]$ deskontu faktorea da.

Ingurunearen egoerak (S): jokoaren egoerak predikatu logikoak dira. Predikatuek entitateen arteko erlazioak definitzen dituzte. 3.3 irudiko jokoaren hasierako egoera honela bezala definituta dago:

$$s_t = \texttt{at(P, bedroom)} \otimes \texttt{at(chest drawer, bedroom)} \otimes \texttt{at(antique trunk, bedroom)} \otimes \texttt{at(bed, bedroom)} \otimes \texttt{closed(chest drawer)} \otimes \texttt{closed(antique trunk)} \otimes \texttt{in(old key, antique trunk)}$$

Amaiera egoera edo irabazi egoerei G deritze eta $G \subset S$ betetzen da. Egoera hauek zeintzuk diren jokoa sortzerakoan erabakitzen da.

Ekintza espazioa (A) uneko egoeran posible diren ekintzek osatzen dute. Adibidearen hasierako egoeran eskuragai dauden ekintzak honako hauek dira:

open(chest drawer)::
$at(P, bedroom) ⊗ at(chest drawer, bedroom) ⊗
closed(chest drawer) ⊸ open(chest drawer)

open(antique trunk)::
$at(P, bedroom) ⊗ at(antique trunk, bedroom) ⊗ closed(antique trunk)
⊸ open(antique trunk)

Predikatu logikoak uler daitezen hona hemen notazioaren azalpena: letra larri etzanaz dauden hizkiek mota zehatzeko objektuak ordezkatzen dute (*K: key* (giltza), *S: supporter* (euskarria), *C: container* (edukiontzia) eta *R: room* (gela)). P, I hizkiek jokalaria eta inbentarioa dira, hurrenez hurren. ⊸ sinboloa inplikazio logikoaren sinboloa da. Inplikazioan ezkerreko baliabideak kontsumitzen dira eskumakoak sor daitezen. Azkenik, klausula batek $ sinboloa izanez gero, eskumako aldera pasatuko dela adieraziko da.

Egoera Trantsiziozko Funtzioa (T) logika linearra erabiliz definitzen da. Erregela logikoak ezagutza basean gordetzen dira eta jokoaren ekintza posibleak definitzen dituzte. Adibidearen kasuan honako hauek dira:

open (C):: $at(P, R) ⊗ $at(C, R) ⊗ closed(C) ⊸ open(C)

close (C):: $at(P, R) ⊗ $at(C, R) ⊗ open(C) ⊸ closed(C)

take (K, C):: $at(P, R) ⊗ $at(C, R) ⊗ $open(C) ⊗ in(K, C) ⊸ in(K, I)

$$\textit{take (K, S)}:: \$\texttt{at}(\texttt{P}, R) \otimes \$\texttt{at}(S, R) \otimes \texttt{on}(K, S) \multimap \texttt{in}(K, \texttt{I})$$

$$\textit{insert (K, C)}:: \$\texttt{at}(\texttt{P}, R) \otimes \$\texttt{at}(C, R) \otimes \$\texttt{open}(C) \otimes \texttt{in}(K, \texttt{I}) \multimap \texttt{in}(K, C)$$

$$\textit{put (K, S)}:: \$\texttt{at}(\texttt{P}, R) \otimes \$\texttt{at}(S, R) \otimes \texttt{in}(K, \texttt{I}) \multimap \texttt{on}(K, S)$$

Sari-funtzioa (R) TextWorld-en sortutako jokoetan sari positiboak baino ez ematen dira. Jokalariaren helburua sarietatik espero den deskontatutako zenbatekoa $E[\sum_t \gamma^t r_t]$ non $r_t = R(s_t, a_t)$ maximizatzea da.

3.2.6 TextWorld eta beste inguruneak

Esan bezala, TextWorld-ek beste ingurunetan sortutako jokoetara jokatzeko aukera ematen du, esaterako, *Zork1* ingurunea jokatzeko presta dezake. Horretarako, *Frotz*[4] izeneko interpretatzailea erabili daiteke, (.z5, .z8) motako fitxategientzako eta *Git-Glulx*[5] interpretatzailea .ulx fitxategientzako.

[4] https://davidgriffith.gitlab.io/frotz/
[5] https://github.com/DavidKinder/Git

4. KAPITULUA

Testu-jokoetan jokatzeko agentea

Atal honetan sistemaren xehetasunak, arkitektura eta inplementazioa azalduta daude.

4.1 Sistemaren arkitektura

Arkitektura ulertzeko behar diren kontzeptuak 2.3 atalean azalduta daude. Erabilitako sistema neurona-sare batean datza, *embedding* geruza batez, GRU motako neuronez eta geruza lineal batez osatuta. 2.3.3 azpiatalean azaldutako atentzio-mekanismoa moldatu da, komandoetan aplikatu ahal izateko.

Agenteak gelaren deskribapena (*look* komandoaren emaitza), jokalariaren inbentarioko elementuak (*inventory* komandoaren emaitza) eta jokoaren narrazioa (aurreko komandoa burutu osteko erantzuna) sarrera moduan jasoko ditu (`obs`). GRU motako kodetzaile batek (`encoder_GRU`) sarrerako testua aztertuko du eta uneko egoeraren ezaugarriak zeintzuk diren kalkulatuko du. Ondoren, beste GRU motako kodetzaile bati (`state_GRU`) ezaugarri horiek pasako dizkio, egoera historia bezala funtzionatuko duena, episodio osoan zehar.

Zein komando burutu erabakitzeko, agenteak onargarriak diren komandoak jasoko ditu (`commands`), TextWorld-en eskutik. Komando bakoitzari puntuazio bat esleituko zaio, uneko egoera kodetuaren ezkutuko egoera erabiliz. Horretarako, beste GRU kodetzaile bat (`cmd_encoder_GRU`) komandoen kodeketaz arduratuko da. Ondoren, geruza lineal sinple batek (`critic`) puntuazio bat esleituko dio komando bakoitzaren eta uneko egoerako ezkutuko egoeraren kateaketari. Bestetik, beste geruza lineal bat (`att_cmd`) koman-

doa aukeratzeaz (`action`) arduratuko da. Arkitekturaren diagrama bat ikus daiteke 4.1 irudian.

Hitzen esanahi-bektoreak ez daude aurre-entrenaturik, hau da, ez da erabiltzen. Beraz, esanahi-bektoreak zerotik entrenatzen dira, sistema osoa entrenatu ahala.

4.1.1 Arkitekturaren xehetasunak

Lehenik eta behin, neurona-sarearen sarrerako tamaina eta ezkutuko tamaina zein izango den definitu da. Sarrera bezala, 1000 tamainako hiztegia erabili da eta ezkutuko geruzen tamaina 128 da. Azkenik, irteerako tamaina 1 izango da, bai Actor zatian, bai Critic zatian, izan ere, lehenengoak komando bat emango du eta bigarrenak komando horren puntuazioa. Komandoen hautaketa egiteko atentzioa erabili da, komando bakoitzaren hitzak hobeto baloratzeko.

4.2 Entrenamendua

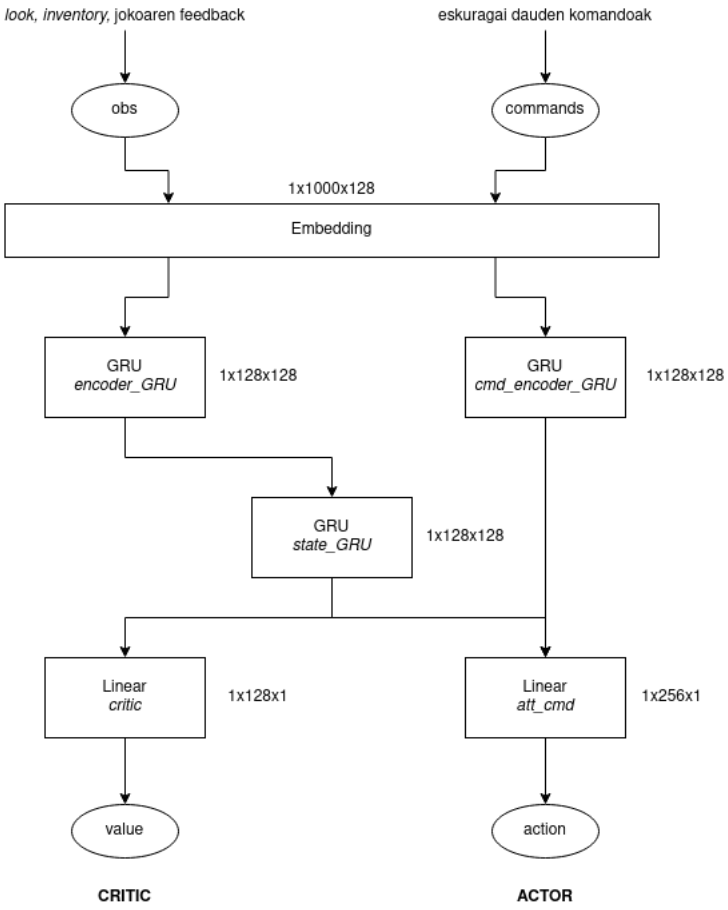
Atal honetan entrenamenduaren nondik norakoak azaltzen dira. Alde batetik, erabilitako parametroen azalpena eta haien balioak agertzen dira. Bestetik, entrenamendu algoritmoaren azalpena egiten da, portaera osoa azalduz. Azkenik, erabilitako optimizatzailea zein den azaltzen da.

4.2.1 Entrenamendu-algoritmoa

Agentea A2C (ikusi 2.2.1 azpiatala) erabiliz entrenatu da. Orain arte azaldutakoa Actor zatiari dagokio. Hala ere, Critic-ak geruza berak erabiltzen ditu azken geruza izan ezik. Kasu honetan, beste geruza lineal bat erabiltzen da uneko egoeraren balorazio bat (`value`) egiteko. Entrenamenduan erabilitako algoritmoa 4. algoritmoan agertzen da.

Azkenik, 2.1 atalean azaldu den bezala, errefortzu bidezko ikasketako agenteek esplorazioa eta ustiapenaren arteko elkarrekintza jorratu behar dute. Kasu honetan, esplorazioa sustatzeko entropiaren erregularizazioa erabiltzen da.

Errefortzu bidezko ikasketan entropia definitzeko informazio-teoriatik (Shannon, 1948) erauzi da:



4.1 irudia: Arkitekturaren egitura. Goiko elementuak sarrera elementuak dira (*obs*, *commands*). Elementu horiek *Embedding* geruzatik pasako dira haien kodeketa burutzeko. Ondoren, bakoitzak behar duen prozesua garatuko da. Jokoaren behaketak (*obs*) ezkerreko GRU kodetzailetik (*encoder_GRU*) pasatuko dira hurrengo GRU kodetzailera (*state_GRU*). Bestetik, komandoak eskumako kodetzailera (*cmd_encoder_GRU*) pasatuko dira eta behin kodeketa burutu den, atentzioa aplikatuko zaie, geruza linealaren irteera balioei *softmax* funtzioa aplikatuz. Beraz, balio altuena duen komandoa izango da Actor-aren irteera balioa (*action*). Azkenik, Critic-ak balioa kalkulatuko du egoera ezkutuko balioa erabiliz.

$$H(X) = -\sum_{x \in X} P(x) \log P(x) \tag{4.1}$$

non entropia aldagai multzo diskretu baten aldagai zehatz baterako kalkulatzen den. Errefortzu bidezko ikasketan $\pi(a|s_t)$ politikarena kalkulatzen da:

$$H(\pi(\cdot|s_t)) = -\sum_{a \in A} \pi(a|s_t) \log \pi(a|s_t) \tag{4.2}$$

Agentea bere politika ikasten dagoenean eta ekintza batek sari positiboa bueltatzen duenean, baliteke agenteak etorkizunean ekintza hori bera beti erabiltzea, sari positiboa sortu duela badakielako. Aitzitik, gerta liteke ekintza hobe bat egotea baina agenteak inoiz ez burutzea dakiena bakarrik ustiatuko duelako. Hori dela eta, agentea optimo lokal batean harrapatuta gera liteke, optimo globala lortu beharrean.

Entropiari esker, esplorazioa sustatu daiteke eta ondorioz, optimo lokal batean geratzea saihestu daiteke. Helburu hau garatzeko, RL-aren helburuaren balioa handitu daiteke, entropia gehituz (Ziebart, 2010; Haarnoja et al., 2017).

4.2.2 Entrenamenduko parametroak

Hauek dira entrenamendua burutzeko erabili diren parametroak:

- Eguneratze-maiztasuna: neurona-sarearen pisuak noiz eguneratzen diren. **Balioa**: 10 ekintzaz behin.
- γ (deskontu-faktorea): 3.2.5 azpiatalean azaldutako deskontu faktorea. **Balioa**: 0,9.
- Jakinarazpen-maiztasuna: neurona-sareko balioak erabiltzaileari noiz jakinarazten zaizkion. **Balioa**: 1000 ekintzaz behin.

4.2.3 Optimizazio-algoritmoa

Entrenamenduan erabili den optimizazio-algoritmoa Adam (Kingma and Ba, 2014) izan da. Adam gradiente jaitsiera estokastikoaren (SGD) (Kiefer et al., 1952) aldaera bat da. SGD-en ikasketa-tasa bakarra dago parametro guztientzako. Adam optimizatzailearen kasuan, ikasketa-tasa desberdina mantentzen da neurona-sareko pisu bakoitzerako. Egileek

Algoritmoa 4 A2C (Advantage Actor Critic)

θ, w politika eta pisuak ausaz hasieratu
$\alpha \leftarrow 0.0003$ ikasketa-tasa hasieratu
E : episodio kopurua
t_{max} : pauso kopurua
for $e = 1 \ldots E$ **do**
  $t \leftarrow 1$
  Ingurunea berrabiarazi eta $s_1 \leftarrow$ hasierako egoera
  $entropy \leftarrow 0$
  repeat
    Balioa eta politika kalkulatu $Q_{val}, \pi \leftarrow \text{A2C}_\theta(s_t)$
    Hurrengo ekintza lortu $a_t \sim \pi(a_t|s_t)$
    Entropia eguneratu $entropy \leftarrow entropy - \sum_i \pi_i * \log \pi_i$
    Probabilitateen logaritmoa gorde LP[t] $\leftarrow log(\pi_{a_t})$
    Saria eta hurrengo egoera lortu $r_t, s_{t+1} \leftarrow \text{P}(a_t, s_t)$
    Balioa eta saria gorde $V[t] \leftarrow Q_{val}, R[t] \leftarrow r_t$
    $t \leftarrow t + 1$
  until bukaerako s_t egoera **or** $t = t_{max}$
  Balioa kalkulatu $Q_{vals}[t], \_ \leftarrow \text{A2C}_\theta(s_t)$
  for $i \in \{t-1 \ldots 1\}$ **do**
    $Q_{vals}[i] \leftarrow R[i] + \gamma * Q_{vals}[i+1]$
  end for
  $\text{loss}_{actor} \leftarrow mean(-LP * (Q_{vals} - V))$
  $\text{loss}_{critic} \leftarrow 0.5 * mean((Q_{vals} - V)^2)$
  $loss \leftarrow loss_{actor} + loss_{critic} + 0.001 * entropy$
  Politika eguneratu $\theta \leftarrow \theta + \alpha \nabla_\theta loss$
end for

Adam beste bi algoritmoen konbinaketa bezala definitzen dute: AdaGrad (Duchi et al., 2011) eta RMSProp[1]. Erabilitako ikasketa-tasa 0,00003 izan da.

4.2.4 Gradient clipping

Entrenamenduan zehar gradienteak balio handiak har ditzake, eta horren ondorioz sarearen parametroen eguneraketan aldaketa handiegiak ager daitezke, entrenamendua ezegonkortuz.

Hori ekiditeko *gradient clipping* teknika erabili da. Teknika honi esker gradientearen balio batetik aurrera balioa mugatu daiteke. Entrenamenduan teknika hau aplikatu da gradientearen normak 40 balioa gainditzean.

4.2.5 Padding

Normalean, sarrerako esaldi guztiek ez dute luzera bera, baina kodetzeko erabiltzen diren bektoreak, ordea, luzera berekoak izan behar dira. Arazo hori ekiditeko, *padding* teknika erabiltzen da, non hutsuneak dauden lekuetan <PAD> ikurra gehitzen den.

[1] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

5. KAPITULUA

Ebaluazioa eta emaitzak

Kapitulu honetan egindako esperimentuak eta haien emaitzak azaltzen dira. Lehenik eta behin, zenbait esperimentutan lortutako emaitzak agertzen dira, haien xehetasunak zehaztuz. Bestetik, sistemak ikasitakoa ebaluatzen da.

5.1 Esperimentuen diseinua

Esperimentuak burutzeko TextWorld-eko *simple* motako (3.2.2) jokoak erabili dira. Bi esperimentu burutu egin dira, jokoen ezaugarri desberdinekin. Lehenengoan, jokoaren helburua pausoz pauso azalduta dago, (*goal detailed*). Bigarrenean, ordea, jokoaren helburua esaldi bakarrean azaltzen da, burutu behar den azken ekintza, hain zuzen ere (*goal brief*). Esperimentu bietan sariak ekintza zuzen bat burutzen den bakoitzean ematen dira (*rewards dense*). Bi esperimentuetan bi multzo egon dira: entrenamendu multzoa eta test multzoa. Lehenengoan 100 joko egon dira eta bigarrenean, aldiz, 20.

Esperimentuko joko gehienak sortzeko 3.2.3 azpiataleko lehenengo metodoa erabili da. Dena den, joko bereziren bat sortu nahi izan denean, hau da, jokoak helburu edota egitura zehatz bat izan zezan, azpiatal berean aipatutako *GameMaker* tresna erabili da.

Erabilitako jokoen egitura berdina da, hau da, gelen kopuru berdina dago, posizio berean eta haien arteko konexioak (ateak, korridoreak) berdinak dira. Horrez gain, geletan dauden objektuak berdinak dira, janaria izan ezik. Janariaren kasuan, objektuek hiru kokapen

desberdin izan ditzakete: sukaldea, egongela edota lorategia. Sukaldean daudenean, hozkailu barruan egongo dira; egongelan, aldiz, sofan eta lorategian, zoruan bertan.

Tauletan agertzen diren emaitza guztiak batez-besteko balioak dira, 100 pauso maximoko 10 episodiotan zehar. Esperimenturen batean parametro horiek aldatu izan badira adieraziko da.

5.2 Esperimentazioa eta emaitzak

Sistemaren jokaera ikusteko asmoz, zenbait proba desberdin egin dira, atal honetan aurkeztutakoak.

5.2.1 Joko bakarreko esperimentua

Hasteko, ausazko agente bat erabili da *baseline* gisa. Agente honek unean eskuragai dauden komandoen artean ausaz bat aukeratuko du, inolako irizpiderik gabe. Agente honen emaitzak hiru jokoetan, bakoitza sari banaketa mota batekin, honako hauek izan dira:

Jokoaren sari banaketa	**Pausoak**	**Puntuazioa**
Sarri (*dense*)	100	4.2 / 10
Orekatu (*balanced*)	100	0.7 / 4
Eskas (*sparse*)	100	0.0 / 1

5.1 taula: Ausazko agentearen emaitzak

Ikusi daitekeenez, lortutako emaitzak nahiko eskasak dira, ausazko agente batekin bat etortzen direnak. Aipatzekoa da nola emaitzak okertzen doazen sarien maiztasuna jaitsi ahala, izan ere, sari maiztasuna handia denean, aukera gehiago daude punturen bat lortzeko, puntuak ekintza zuzen bat burutzean lortzen delako. Bestetik, azken kasuan, ez da inolako punturik lortzen puntuak jokoaren helburua lortzen denean soilik lortzen direlako.

Bestetik, 4.1 atalean azaldutako sistema erabiltzen duen agente neuronal bat sortzean eta 500 episodioz sariak sarri ematen dituen joko batean entrenatu eta testatu ostean honako emaitza hauek lortzen dira:

Agenteak entrenatu gabe jokatzen duenean, ausazko agentearen emaitzen antzekoak lortzen ditu, baina behin entrenatuta egon emaitza nahiko onak lortzen ditu. Dena den, agentea joko bakarrean entrenatu da, hortaz, ez da arraroa emaitza hain altuak lortu izana.

Agentea	Pausoak	Puntuazioa
Ausazko agentea	100	4.2 / 10
Agente neuronala (entrenamendurik gabe)	95.4	4.5 / 10
Agente neuronala (entrenatuta)	77.9	8.6 / 10

5.2 taula: Ausazko agentearen eta agente neuronalaren emaitzak, jokoaren sari banaketa *dense* denean

Agente horrek emaitza oso onak lortu ditu entrenatutako jokoan. Baina agente horri beste joko bat jarriz gero, emaitzak nabarmen okertzen dira (5.3 taula).

Agentea	Pausoak	Puntuazioa
Ausazko agentea	100	3.9 / 8
Agente neuronala (entrenatuta)	97.1	4.4 / 8

5.3 taula: Ausazko agentearen eta agente neuronalaren emaitzak joko berri batean

Agentearen ez-eraginkortasuna agerian geratzen da joko berri batean jokatzen jartzen denean.

5.2.2 Helburu zehatzeko joko-sorta

Aurreko emaitzak hobetzeko asmoz, agente berri bat 100 jokotan entrenatu da, kasu honetan joko bakoitzeko 5 episodiotan. Joko hauek *simple* motako jokoak dira, helburu zehatza dutenak non sariak ekintza bat ondo egiten den bakoitzean ematen diren.

Entrenatu ostean, ezaugarri bereko 20 jokotan testatu egin dira, kasu honetan, 10 episodiotan zehar (5.4 taula). Ikusten denez, emaitzak oso onak dira eta sistema neuronala joko ezberdinak ikasteko gai dela erakusten du.

Agentea	Pausoak	Puntuazioa
Ausazko agentea	100	0.4 / 1
Agente neuronala	88.4	0.8 / 1

5.4 taula: Ausazko agentearen eta agente neuronalaren emaitzak helburu zehatzeko 20 jokotan

5.2.3 Helburu laburreko joko-sorta

Sistemaren irismena neurtzeko asmoz, helburu laburra duten jokoetan agente berri bat ebaluatu nahi izan da. Horretarako, goiko esperimentuaren irizpide berak jarraitu dira eta hauek izan dira lortutako emaitzak:

Agentea	**Pausoak**	**Puntuazioa**
Ausazko agentea	99.5	0.5 / 1
Agente neuronala	91.1	0.8 / 1

5.5 taula: Ausazko agentearen eta agente neuronalaren emaitzak helburu laburreko 20 jokotan

Ausazko agentearen emaitzaren puntuazioa 0.1ean hobetu da, baina hori kointzidentzia hutsa baino ez da, izan ere, esan bezala, ausazko agenteak ez du jarraitzen inolako irizpiderik. Agente neuronalari dagokionez, emaitzak aurrekoan bezala mantendu dira, pauso kopuruan gehikuntza txiki bat baztertuz.

5.2.4 Egitura desberdina duen jokoa

Helburu laburreko 100 jokoetan entrenatu den sistemak mundu eta hizkuntzari buruz ikasi duena ikusteko asmoz, bi motako esperimentuak diseinatu dira. Lehenengoan, jokoaren egitura aldatu egin da, agentearen portaera ingurune berrian ikusteko. Horretarako, jokoen gelen kokapena aldatu da, geleko objektuen kokapena mantenduz. Jokoaren helburua oilaskoa hozkailutik hartzea eta suan jartzea izan da, era zehatzean azalduta (eman beharreko pauso guztiak). Agentea 50 episodiotan jokatzen testatu da (5.6 taula).

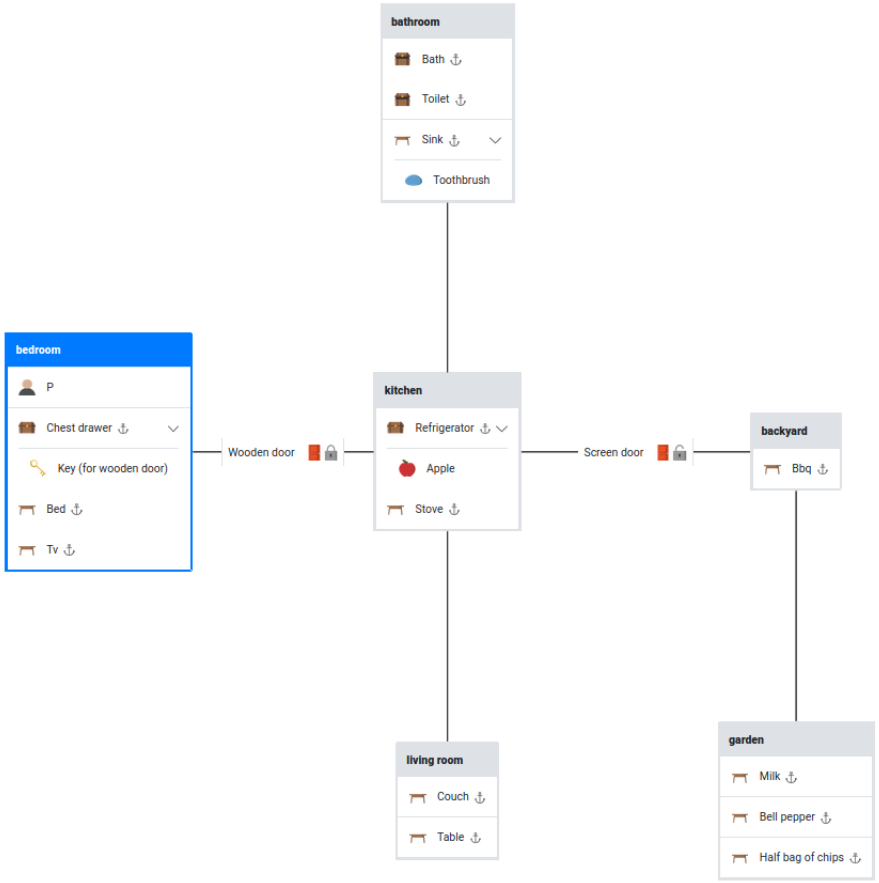
Pausoak	**Puntuazioa**
99.2	1.6 / 6

5.6 taula: Agentearen emaitzak egitura desberdina duen joko baten, 50 episodiotan zehar

Emaitzak ikusita, jokoaren egitura aldatzean agenteak munduan aritzen ez dakiela ondoriozta daiteke, izan ere, bataz-beste ez ditu ezta bi ekintza ondo burutzen.

5.2.5 Neurrira egindako joko-sorta

Azkenik, entrenamenduko jokoen egitura bera duten joko batzuk (5.1 irudia) sortu egin dira, bakoitza helburu zehatz batekin, munduko gela batera joatea, hain zuzen ere. Siste-



5.1 irudia: Jokoen egitura. Jokalaria logelan (*bedroom*) hasiko da eta ezarritako helburuaren arabera gela batera joan beharko da.

maren irismena ikusteko, 5.2.3 azpiataleko agentea joko hauetan testatu egin da.

Esan bezala, jokoen helburua hasieran adierazitako gela batera joatea da. Lehenengo gelatik irteteko bost komando burutu behar dira beti (komandoren batean parentesiak egonez hautazkoa dela esan nahi du):

1. *open chest drawer* (kutxa zabaldu)
2. *take key (from chest drawer)* ((kutxatik) giltza hartu)
3. *unlock wooden door with key* (egurrezko atea giltza erabiliz ireki)
4. *open wooden door* (egurrezko atea zabaldu)
5. *(go) east* (ekialdera (joan))

Helburua	Helburura heltzeko pausoak	Pauso kopuru maximoa	Pausoak	Puntuazioa	Helburua lortutako aldiak
Sukaldea	5	100	10.1	4.0 / 4	10 / 10
		7	6.8	2.8 / 4	5 / 10
Komuna	6	100	48.7	4.7 / 5	7 / 10
		20	18.3	4.2 / 5	4 / 10
		10	9.7	3.3 / 5	0 / 10
Patioa	7	100	34.6	6.0 / 6	10 / 10
		20	17	4.9 / 6	3 / 10
		10	10	3.9 / 6	0 / 10
Egongela	6	100	43.6	4.8 / 5	7 / 10
		20	17.4	4.0 / 5	4 / 10
		10	9.7	3.7 / 5	0 / 10
Lorategia	8	100	33.7	7.0 / 7	10 / 10
		20	20	4.2 / 7	1 / 10
		10	10	2.8 / 7	0 / 10

5.7 taula: Agentearen emaitzak helburu gelaren eta pauso kopuru maximoaren arabera

Ondoren, gela bakoitzeko emaitzak banan-banan aztertuko dira.

Sukaldea

Behin aurreko urratsak beteta, jokalaria sukaldean (*kitchen*) egongo da, testatutako lehenengo helburua. Agenteak lortutako puntuazioak nahiko onak izan dira helburu hau betetzerako orduan.

Sukaldera heltzeko gutxienez bost pauso beharrezkoak dira. Lehenengo kasuan, gehienez 100 pauso eman daitezkeenean, helburua %100 alditan lortzen da, 10 pausotan. Bestetik, pauso kopuru maximoa 7ra murrizten denean, pauso guztiak agortzen dituela igarri daiteke, baina helburua lortu gabe, izan ere, 3 puntu lortzen ditu. Beste era batera esanda, pauso kopurua %93an murrizten denean, helburua saiakeren %50ekoan lortzen du.

Komuna

Hurrengo helburua komuna (*bathroom*) da. Hona heltzeko aurretik esandako bost pausoak gehi beste bat (*go north*) burutu behar dira.

Kasu honetan, nahiz eta pauso bakar bat gehiago behar, ia 40 pauso gehiago behar ditu helburua lortzeko. Behin sukaldean egonda, ekintza posibleen kopurua hazten da; hasierako gelan irteera bakarra dago, baina sukaldean, ordea, lau irteera posible daude. Helburua

lortzeko 100 pauso maximo izanda, agenteak helburua %70ean lortzen du. Aldiz, pauso kopuru maximoa 20ra jaisten denean, batez-besteko 4 puntu lortzen ditu, baina helburua soilik %40 alditan lortzen du. Azkenik, pauso kopuru maximoa 10 denean, helburua ez du inoiz lortzen.

Patioa

Testatu den hurrengo helburua patioa (*backyard*) izan da. Hona heltzeko sukaldera heltzeko behar diren pausoez gain, beste bi ere burutu egin behar dira:

6. *open screen door* (kristalezko atea ireki)
7. *(go) east* (ekialdera (joan))

Kasu honetan, helburua lortzeko 100 pauso posible daudenean, helburua episodioen %100ean lortzen da. Kopuru hori 20ra arte murrizten denean, helburua 3 episodiotan lortzen da, baina bataz-besteko puntuazioa 5ekoa da, hau da, soilik pauso bat falta zaio helburura heltzeko. Azkenik, pauso kopuru maximoa 10 denean, hau da, bide optimoa baino 3 pauso gehiago eskuragai daudenean, agentea ez da inoiz helburura heltzen. Gainera, batez-besteko puntuazioa ikusiz, sukaldera ez dela heltzen ondoriozta daiteke, izan ere, sukaldera heltzean 4 puntu lortzen dira.

Egongela

Egongelaren (*bedroom*) kasua, komunaren kasuaren antzekoa da. Kasu honetan, behin sukaldean egonda iparraldera jo beharrean, hegoalderantz joan behar da (*go south*). Emaitza hauek komuneko emaitzekin alderatzen badira, balio oso antzekoak aurki daitezke. Alde batetik, pauso kopuruak eta puntuazioa ia berdinak dira, bariantza txiki batekin. Bestetik, helburua lortzen den aldi kopurua berdina da hiru kasuetan. Aipatu den bezala, entrenamenduko jokoetan agenteak ez dauka zertan hemendik igaro, hortaz, honako gela batera igarotzean ez du jasoko inolako saririk.

Lorategia

Azkenik, agentea lorategira (*garden*) heltzen saiatu da. Hona heltzeko patiora heltzeko burutu behar diren komandoez gain, azken bat ere bete behar da, hegoaldera jo behar da (*go south*), hain zuzen ere. Hona heltzeko, beste edozein gelara heltzeko baina pauso gehiago bete behar dira, zortzi, hain zuzen ere. Hemen, patioaren kasuan bezala, 100

pauso maximo eskuragai daudenean, helburua beti lortzen da, bataz-besteko 34 pauso ingurutan. Baina, kopuru hori 20ra murriztean, pauso guztiak agortzen ditu eta helburua soilik behin lortzen du. Pauso kopuru maximoa 10 denean, aldiz, 3 inguruko puntuazioa lortzen du eta helburua ez du inoiz lortzen.

5.3 Analisi kualitatiboa

5.2.5 azpiataleko emaitzak kuantitatiboki aztertzeaz gain analisi kualitatiboa ere egin da. Horretarako, test jokaldi batzuen transkripzioak aztertu dira. Esaterako, sukaldeko eta lorategiko jokaldiak azalduko dira.

Sukaldea

Sukaldeko jokaldien transkripzioak aztertu eta gero, lehenengo lau ekintzak ia beti burutzen direla egiazta daiteke, bai 100 pausoko jokoan bai 7 pausokoan. Bigarren kasuan, behin atea zabalik dagoela, egiteke dagoen bakarra ekialdera joatea da, baina agenteak berehala ez duenez burutzen, pausoak agortzen zaizkio helburura heldu gabe. Dena den, lau pauso horiek beti bete behar direnez, ezin da munduaren ikasketaren ondoriorik atera.

Lorategia

Lorategiaren kasuan, hiru test egon dira. Sukaldearen kasuan bezala, lehenengo lau ekintzak ia beti burutzen dira. 100 pauso maximo eskuragai daudenean, helburua beti lortzen dela ikusi da. Transkripzioak aztertuz, helburura heldu baino lehen agentea mundutik bueltaka dabilela ikus daiteke. Izan ere, lehenengo gelatik irteteko 10 bat pauso behar ditu, hortaz, ulertzekoa da azken bi kasutan, 20 eta 10 pauso eskuragai dituenean, emaitza hain txarrak lortzea. Lehenengo gelatik ateratzen bada, mundutik bueltaka aritzen da, hau da, ez du helburuko bidea zuzenean hartzen. Honekin topologia ez duela ikasi ondoriozta daiteke.

5.4 Eztabaida

Tauletako balioak ikusiz, agenteari hurbileko geletara iristea urrunekoetara baino gehiago kostatzen zaiola igarri daiteke. Esaterako, komunera eta egongelara ez da episodioen %100etan iristen, baina patiora eta lorategira, ordea, bai. Entrenamenduko jokoak begiratuz, jokoaren helburua janari zehatz bat aurkitzea eta prestatzea izaten da. Normalean,

janari hori sukaldean edota lorategian kokatzen da, hortaz, ez da harritzekoa emaitza hoberenak gela horietara joan behar denean lortzea. Patioaren kasuan, lorategira joateko ezinbestekoa da hortik igarotzea, beraz, emaitza onak ere izango ditu.

Beraz, sistemak munduaren topologia ez duela ikasi ondoriozta daiteke. Horrela izan balitz, ezarritako helburuak burutzeko gai izan beharko litzateke. Horren ordez, entrenamendu jokoen helburua lortzen ikasi du, horretarako behar diren komandoak betez, munduaren egiturari begiratu gabe.

6. KAPITULUA

Ondorioak eta etorkizunerako lana

Kapitulu honetan proiektuan zehar lortutako ondorioak aztertuko dira. Ondorio horien arabera etorkizunean proiektuari jarraipena emateko edota hobetzeko eman daitezkeen urratsak ere azaltzen dira.

6.1 Ondorioak

Proiektuan zehaztutako helburuak bete dira, izan ere, sistemak ikasitakoa ebaluatzea lortu da. Aurreko kapituluan azaldu den bezala, sistema ez da gai izan munduaren topologia ikasteko. Aitzitik, sistemak benetan ikasi duena entrenamendu jokoen helburua lortzeko burutu behar diren komandoak zeintzuk diren eta zein ordenatan bete behar dituen da. Horrek motibatzen du beste arkitektura baten beharra, munduaren egitura ikastea ahalbidetzen duena.

Bestalde, RL ikasketa paradigma ulertzeko beharrezko diren kontzeptuak aztertu dira, aipagarrienak, Markov-en erabakitze-prozesuak eta Actor-Critic algoritmoak. Horrez gain, neurona-sareekin ere lan egin da. Ikasketa sakona landu da, errefortzu bidezko ikasketarekin konbinatzeko behar diren elementuak ikertuz, besteak beste, atentzio-mekanismoak, testu-joko munduen errepresentazioen kodeketa, etab.

Testu-jokoei dagokionez, TextWorld inguruneko funtzionamendua ulertzea lortu da, baita dituen hainbat ezaugarri ere. Dena den, oraindik bereizgarri ugari geratzen dira ikusi gabe.

Bestetik, lengoaia naturala ikasteko testuan oinarritutako jokoen onurak zeintzuk diren ikusi da.

Azkenik, artearen egoeran egondako hainbat sistema ikertu dira, bakoitzaren indarguneak eta ahulguneak zeintzuk diren ikusiz. Egin berri diren lan batzuk (Ammanabrolu and Hausknecht, 2020; Ammanabrolu and Riedl, 2019) ikusiz, ezagutza-grafoak gero eta garrantzia handiagoa lortzen ari direla egiazta daiteke. Ikerketa-lerro hau hurrengo atalean aipatzen da.

6.2 Etorkizunerako lana

Esan den bezala, sistemari *bias* bat sartzeko beharra agerian geratu da. Horretarako, ezagutza-grafo izeneko egitura bat erabil daiteke, non munduko ezaugarriak era egituratuan gordetzen diren, nodo eta ertzen bidez.

Aipatutako GATA sistema (Adhikari et al., 2020) mota honetako grafo bat inplementatzen du eta munduaren kodeketa grafo kodetzaile bat erabiliz egiten du. Sistema honen jokaera TextWorld ingurunean interesgarria izan liteke, autoreek lortutako emaitzak nahiko itxaropentsuak baitirudite. Sistema hau martxan jartzeko, dezente denbora behar da, izan ere, aurkeztutako artikuluaren arabera, 18 egun eman du aurre-entrenamenduak oso GPU ahaltsu baten.

Ildo berean, zentzuz jokatzen duten sistemek (Murugesan et al., 2020) etorkizun handiko aukerak ere dirudite. Sistema hauek ezagutza-grafotan oinarritzeaz gain, kanpoko ezagutzaz ere baliatzen dira, horrela, zentuzko ekintzak burutu ditzakete.

Azkenik, beste aukera bat ezagutza-grafoa galderen bidez sortzen duen sistema ikertzea izan liteke. Sistema honek, Q*BERT izenekoa (Ammanabrolu et al., 2020), galderak erabiltzen ditu grafoa eraikitzeko, nabarmeneko eraginkortasuna lortuz.

Bibliografia

Adhikari, A., Yuan, X., Côté, M.-A., Zelinka, M., Rondeau, M.-A., Laroche, R., Poupart, P., Tang, J., Trischler, A., and Hamilton, W. L. (2020). Learning dynamic knowledge graphs to generalize on text-based games. *CoRR*, abs/2002.09127.

Ammanabrolu, P. and Hausknecht, M. (2020). Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Ammanabrolu, P. and Riedl, M. (2019). Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565, Minneapolis, Minnesota. Association for Computational Linguistics.

Ammanabrolu, P., Tien, E., Hausknecht, M., and Riedl, M. O. (2020). How to avoid being eaten by a grue: Structured exploration strategies for textual worlds.

Angeli, G., Johnson Premkumar, M. J., and Manning, C. D. (2015). Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Atkinson, T., Baier, H., Copplestone, T., Devlin, S., and Swan, J. (2019). The text-based adventure ai competition. *IEEE Transactions on Games*, 11(3):260–266.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate.

Bellman, R. (1957). A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

Bellman, R. (1958). Dynamic programming and stochastic control processes. *Information and Control*, 1(3):228 – 239.

Blank, M. and Lebling, D. (1980). Zork i: The great underground empire.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.

Côté, M.-A., Kádár, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Tao, R. Y., Hausknecht, M., Asri, L. E., Adada, M., Tay, W., and Trischler, A. (2018). Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies.

Hasselt, H. V. (2010). Double q-learning. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc.

Hasselt, H. V., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning.

Hausknecht, M., Ammanabrolu, P., Côté, M.-A., and Yuan, X. (2019). Interactive fiction games: A colossal adventure.

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. (2016). Deep reinforcement learning with a natural language action space. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

Kiefer, J., Wolfowitz, J., et al. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Markov, A. (1954). *Theory of Algorithms*. TT 60-51085. Academy of Sciences of the USSR.

McClelland, J. L. and Rumelhart, D. E. (1988). Explorations in parallel distributed processing: a handbook of models, programs, and exercises. *American Journal of Psychology*, 102:435.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Murugesan, K., Atzeni, M., Shukla, P., Sachan, M., Kapanipathi, P., and Talamadupula, K. (2020). Enhancing text-based reinforcement learning agents with commonsense knowledge.

Narasimhan, K., Kulkarni, T., and Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Parisotto, E. and Salakhutdinov, R. (2017). Neural map: Structured memory for deep reinforcement learning.

Pearson, K. (1905). The problem of the random walk. *Nature*, 72(1867):342–342.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sarkar, A. (2018). A brandom-ian view of reinforcement learning towards strong-ai.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Ziebart, B. D. (2010). Modeling purposeful adaptive behavior with the principle of maximum causal entropy.

Eranskinak

A. ERANSKINA

Proiektuaren Helburuen Dokumentua

Atal honetan proiektuaren helburuen dokumentua aurkezten da, proiektuaren deskribapena, plangintza, lan-metodologia, arrisku-plana eta bestelakoak azalduz.

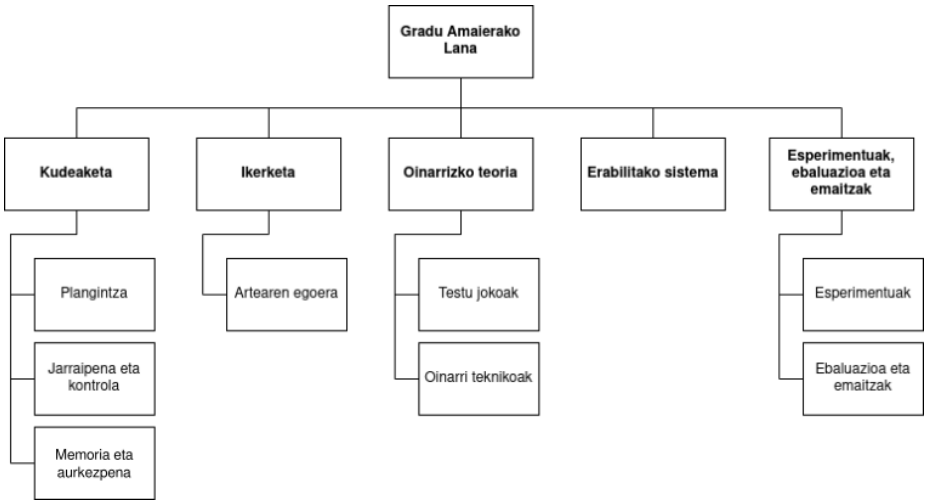
A.1 Proiektuaren deskribapena eta helburuak

Proiektu honetan errefortzu bidezko ikasketa erabili nahi da lengoaia naturalen ikasketa garatzeko. Horretarako, testu-jokoak erabili dira ikasketa paradigma honetan sakontzeko. Testu-jokoak espazio konbinatorial eta konposizionalak bezala interpreta daitezke, beraz, ingurune ezin hobeak dira lengoaia naturalaren ikasketa gauzatzeko. Proiektuaren helburua erabiliko den sistemak TextWorld (Côté et al., 2018) ingurunean errealitatea eta hizkuntzaren arteko zati konkretu bat RL bidez ikasi duen ala ez aztertzea da.

A.2 Proiektuaren plangintza

A.2.1 LDE diagrama

Proiektuan zehar burutu beharreko lanaren deskonposaketa egiteko Lanaren Deskonposaketa Egitura (LDE) diagrama erabili da.



A.1 irudia: LDE diagrama

A.2.2 Lan-paketeak

LDE diagraman definitutako lan-paketeen deskribapena azaltzen da hemen.

Plangintza

Ataza honetan proiektuaren planifikazioa garatu da. Bertan, proiektuaren helburuak, burutu beharreko atazak, emangarriak zeintzuk diren eta lan metodologia zehaztu dira.

Jarraipena eta kontrola

Ataza honetan proiektuko helburu eta mugarri guztiak betetzeko egin beharrekoa azaltzen da. Horretarako, astero jarraipen bilerak burutuko dira.

Memoria eta aurkezpena

Ataza honetan proiektuaren memoria eta aurkezpena garatuko dira.

- **Memoria**: proiektuaren inguruko xehetasun guztiak dituen dokumentua da.
- **Aurkezpena**: proiektuaren defentsan erabiliko den aurkezpena.

Artearen egoera

Ataza honetan momentura arte dauden sistemak aztertuko dira, metodo desberdinak aztertuz eta horiek testu-jokoetan nola aritzen diren ikusiz.

Testu-jokoak

Ataza honetan testu-jokoak zer diren eta nola funtzionatzen duten ikusiko da. Behin ideia orokor bat izanda, TextWorld-en dokumentazioa aztertuko da, proiektua garatzeko beharrezkoa dena uler dadin.

Oinarri teknikoak

Ataza honetan erabilitako sistema ulertzeko behar diren oinarri teknikoak aztertuko dira, haien artean, errefortzu bidezko ikasketa eta neurona-sareak.

Erabilitako sistema

Ataza honetan erabilitako sistema sakoneran aztertuko da, guztiz ulertzeko. Horretarako, erabiltzen duen algoritmoa eta bere funtzionamendua ikertuko da.

Esperimentuak

Ataza honetan planifikatutako esperimentuak burutuko dira. Gerta liteke esperimentu batzuk bertan behera utzi beharra edota planifikatuta ez zeuden esperimentuak burutzea.

Ebaluazioa eta emaitzak

Ataza honetan burututako esperimentuen ebaluazioa eta haien emaitzak ikertuko dira, sistemaren eraginkortasuna ikusteko.

A.2.3 Emangarriak

Proiektuan garatu beharreko emangarriak honako hauek dira:

- Memoria
- Aurkezpena

A.2.4 Mugarriak

Proiektuan zehar garatu beharreko emangarrien mugarriak azaltzen dira hurrengo taulan:

Lan-paketea	Iraupena(ordutan)
Kudeaketa	**125**
Plangintza	5
Jarraipen eta kontrola	20
Memoria eta aurkezpena	100
Artearen egoera	**70**
Oinarrizko teoria	**120**
Testu-jokoak	60
Oinarri teknikoak	60
Erabilitako sistema	**80**
Esperimentuak, ebaluazio eta emaitzak	**60**
Esperimentuak	45
Ebaluazioa eta emaitzak	15
GUZTIRA	**455**

A.1 taula: Lan-pakete bakoitzaren denbora

Emangarria	Entregatze-data
Memoria	2020/06/21
Aurkezpena	2020/06/29-2020/07/10

A.2 taula: Mugarriak

A.3 Lan-metodologia

Proiektu hau hizkuntzaren prozesamenduan aritzen den IXA ikerketa-taldean hasi zen 300 orduko proiektu baten jarraipena izan da.

A.3.1 Bilerak

Proiektuaren jarraipena egiteko asmoz, bilerak astero egingo dira, asteazkenetan goizeko 10:00etan inolako aldaketarik ez badago. Aldaketaren bat egitekotan, posta elektronikoz adostuko da.

A.4 Informazio-sistema

Proiektuaren informazio-sistema 2 azpiegituraz osatuta egongo da:

- *Overleaf* plataforma. Hemen proiektuaren memoriaren garapena egingo da. Hodeian dagoenez, lokalean galtzeko arriskua saihesten da.
- *Google* zerbitzuak. Esperimentuak burutzeko behar diren datuak gordetzeko *Google Drive* plataforma erabiliko da. Esperimentuak burutzeko, aldiz, *Google Colaboratory* erabiliko da, esperimentuak garatzea ahalbidetzen duena ordenagailuko baliabideak kontsumitu gabe.

A.5 Arriskuen kudeaketa

Proiektuaren dimentsioa ikusita, hainbat arrisku eta oztopo egon daitezke proiektu osoaren bizitzan zehar. Hori dela eta, agertu daitezken arriskuak aurreikuspena egitea eta haien prebentzioa egiteko saiakera egin da:

- Ataza batean aurreikusitako denbora baino gehiago irautea eta ondorioz, hurrengo atazak atzeratzea.
- Proiektuaren irismena dela eta, ikasketako entrenamendu parteak denbora handia behar izatea.

Aurreko arriskuak murrizteko asmoz, honako prebentzio neurriak hartzea erabaki da:

- Atazaren baten atzerapenenak dela eta, plangintza malgua izatea erabaki da.
- Exekuzio luzeak egotekotan, beste atazak burutuko dira bitartean.

A.6 Egondako desbiderapenak

Proiektuko hasierako helburua GATA (2.5.2) sistema erabiltzea zen. Baina aurre entrenamendua hasterakoan izugarrizko denbora behar zuela ikusi zen, izan ere, autoreek (Adhikari et al., 2020) entrenamendu denbora aipatzen dute, 18 egun oso GPU eraginkorrak erabiliz. Hori dela eta, hasierako helburua eta erabiliko zen sistema aldatzea erabaki zen.

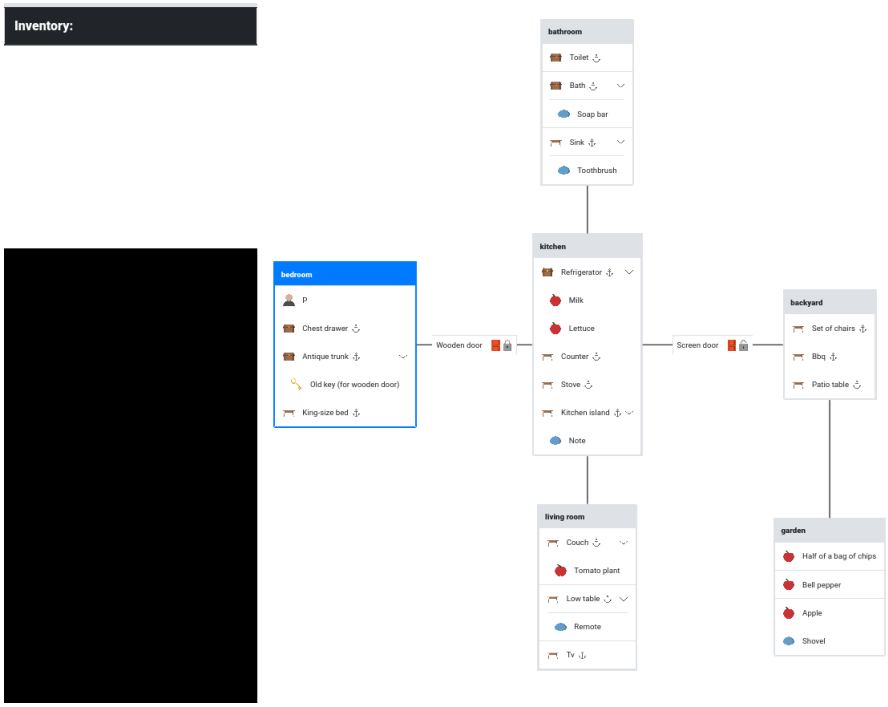
B. ERANSKINA

Joko mota desberdinen adibideak

Atal honetan 3.2.2 azpiatalean aipatutako joko moten adibide bana agertzen da. Joko bakoitzaren adibide bat azaltzen da, munduaren egituraren irudi bat eta jokaldi baten transkripzioa txertatuz. Transkripzioa jokaldi optimoarena izango da, hau da, burutu behar diren ekintzak burutuko dira, akatsik gabe.

B.1 Joko sinplea

```
The dinner is almost ready! It's only missing a grilled lettuce.

-= Bedroom =-
You've just walked into a bedroom.

You smell an awful smell, and follow it to a chest drawer. A closed antique
trunk is right there by you. You can make out a king-size bed. Unfortunately,
there isn't a thing on it.

There is a closed wooden door leading east.

> open antique trunk
You open the antique trunk, revealing an old key.
```



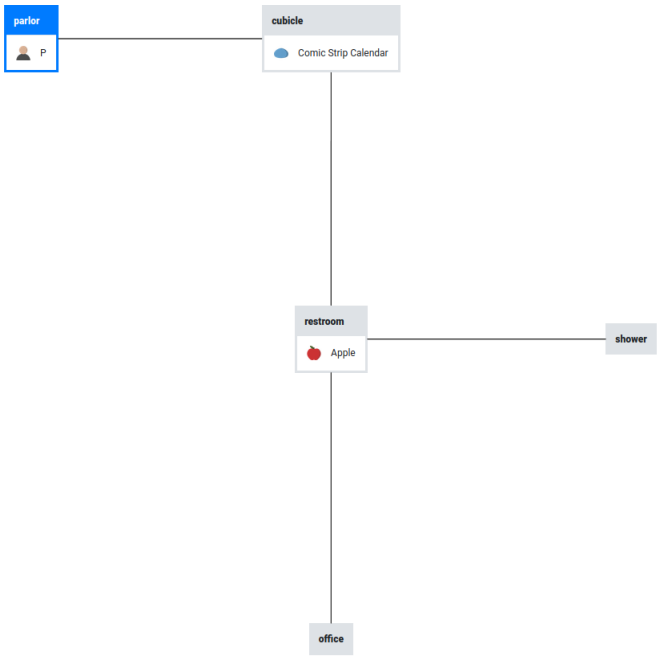
B.1 irudia: *Simple* joko mota, sarriko sariekin (*rewards dense*) eta helburu laburrarekin (*goal brief*).

```
Your score has just gone up by one point.

> take key
You take the old key from the antique trunk.


Your score has just gone up by one point.

> unlock wooden door with key
You unlock wooden door.


Your score has just gone up by one point.

> open wooden door
You open wooden door.


Your score has just gone up by one point.
```

```
> go east

-= Kitchen =-
Well, here we are in a kitchen.

You make out a refrigerator. Oh wow! Is that what I think it is? It is! It's a
counter. The counter is typical. But oh no! there's nothing on this piece of
junk. Oh! Why couldn't there just be stuff on it? You can see a stove. The stove
is typical. But the thing is empty, unfortunately. You see a kitchen island. On
the kitchen island you make out a note.

There is a closed screen door leading east. There is an open wooden door leading
west. You need an unguarded exit? You should try going north. There is an
unblocked exit to the south.


Your score has just gone up by one point.

> open refrigerator
You open the refrigerator, revealing a milk and a lettuce.

> take lettuce
You take the lettuce from the refrigerator.


Your score has just gone up by one point.

> put lettuce on stove
You put the lettuce on the stove.


Your score has just gone up by one point.


                        *** The End ***

You scored 7 out of a possible 7, in 8 turn(s).


Would you like to RESTART, RESTORE a saved game, QUIT or UNDO the last command?
>
```

```
Done after 7 steps. Score 7/7.
```

B.2 Treasure hunter



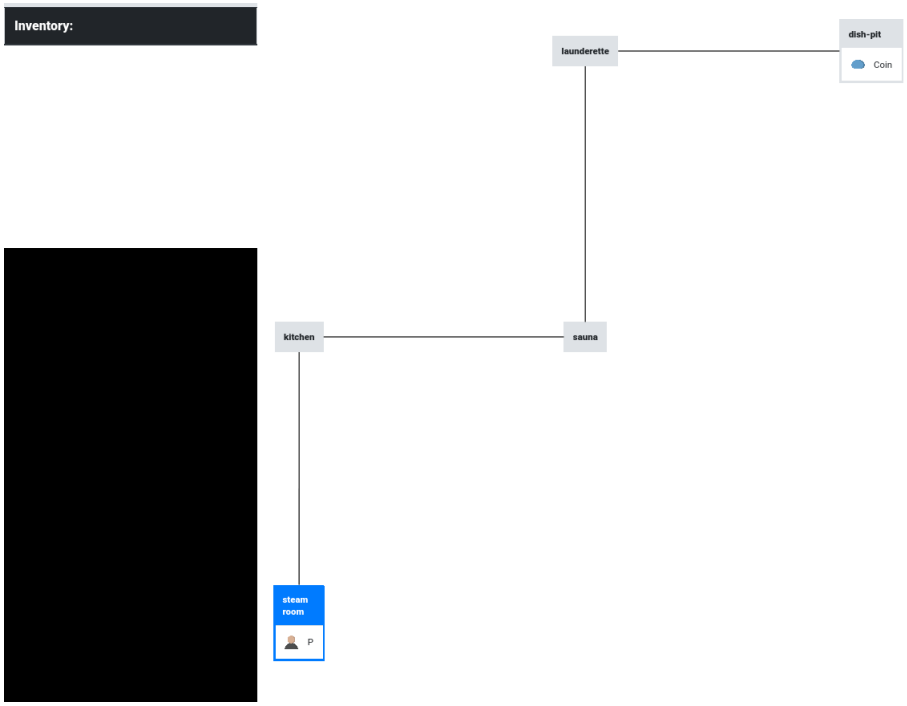
B.2 irudia: 5. mailako *treasure hunter* joko mota.

```
Hey, thanks for coming over to the TextWorld today, there is something I need
you to do for me. Your first objective is to try to head east. And then, go to
the south. Then, retrieve the apple that's in the restroom. And if you do that,
you're the winner!

-= Parlor =-
You find yourself in a parlor. A typical one.



You don't like doors? Why not try going east, that entranceway is unblocked.
```

```
> go east

-= Cubicle =-
You arrive in a cubicle. An ordinary one.



You don't like doors? Why not try going south, that entranceway is unguarded.
You need an unguarded exit? You should try going west.

There is a Comic Strip Calendar on the floor.

> go south

-= Restroom =-
You arrive in a restroom. A normal one.



You don't like doors? Why not try going east, that entranceway is unguarded.
There is an unblocked exit to the north. You need an unguarded exit? You should
try going south.

There is an apple on the floor.

> take apple
You pick up the apple from the ground.



Your score has just gone up by one point.


                              *** The End ***

You scored 1 out of a possible 1, in 4 turn(s).


Would you like to RESTART, RESTORE a saved game, QUIT or UNDO the last command?
>
```

```
Done after None steps. Score 1/1.
```

B.3 Coin collector



B.3 irudia: 5. mailako *coin collector* joko mota.

```
Welcome to TextWorld! Here is how to play! Your first objective is to move
north. That done, make an effort to venture east. After that, go to the north.
If you can accomplish that, try to go east. And then, lift the coin from the
floor of the dish-pit. Got that? Good!

-= Steam Room =-
You find yourself in a steam room. A standard kind of place.



There is an exit to the north. Don't worry, it is unguarded.

> go north
```

```
-= Kitchen =-
You've entered a kitchen.



There is an exit to the east. Don't worry, it is unguarded. You need an
unblocked exit? You should try going south.

> go east

-= Sauna =-
You are in a sauna. A typical kind of place.



There is an exit to the north. Don't worry, it is unguarded. There is an
unblocked exit to the west.

> go north

-= Launderette =-
You've just walked into a launderette. You decide to just list off a complete
list of everything you see in the room, because hey, why not?



You don't like doors? Why not try going east, that entranceway is unblocked.
There is an exit to the south. Don't worry, it is unblocked.

> go east

-= Dish-Pit =-
You are in a dish-pit. A standard one. You begin to take stock of what's here.



There is an unguarded exit to the west.

There is a coin on the floor.

> take coin
```

```
You pick up the coin from the ground.



Your score has just gone up by one point.

                                *** The End ***

You scored 1 out of a possible 1, in 6 turn(s).


Would you like to RESTART, RESTORE a saved game, QUIT or UNDO the last command?
>

Done after 5 steps. Score 1/1.
```

C. ERANSKINA

Joko sinple baten sorkuntza *GameMaker* erabiliz

```
import textworld
from textworld import GameMaker

# GameMaker objektu bat sortu
M = GameMaker()

# munduko gelak sortu
kitchen = M.new_room("kitchen")
livingRoom = M.new_room("living room")
bedroom = M.new_room("bedroom")
bathroom = M.new_room("bathroom")
backyard = M.new_room("backyard")

# gelen arteko korridoreak sortu
kitchen_bedroom = M.connect(kitchen.west, bedroom.east)
kitchen_bathroom = M.connect(kitchen.north, bathroom.south)
kitchen_backyard = M.connect(kitchen.east, backyard.west)
kitchen_livingRoom = M.connect(kitchen.south, livingRoom.north)
backyard_garden = M.connect(backyard.south, garden.north)

# ateak gehitu
wooden_door = M.new_door(kitchen_bedroom, name = "Wooden door") #atea sortu
M.add_fact('locked', wooden_door) #atearen egoera definitu
screen_door = M.new_door(kitchen_backyard, name = 'Screen door')
M.add_fact('unlocked', screen_door)
```

```
26
27 # geletako elementuak sortu
28
29 # bedroom
30 chest_drawer = M.new(type = 'c', name = 'Chest drawer') # elementua sortu
31 M.add_fact('closed', chest_drawer) # itxita dagoela adierazi
32 bedroom.add(chest_drawer) # gelan kokatu
33 antique_trunk = M.new(type = 'c', name = 'Antique trunk')
34 M.add_fact('closed', antique_trunk)
35 bedroom.add(antique_trunk)
36 old_key = M.new(type = 'k', name = 'Old key')
37 M.add_fact('match', old_key, wooden_door) # giltza eta atea lotu
38 antique_trunk.add(old_key) # giltza barruan sartu
39 bed = M.new(type = 's', name = 'King-size bed')
40 bedroom.add(bed)
41
42
43 # kitchen
44 refrigerator = M.new(type = 's', name = 'Refrigerator')
45 M.add_fact('closed', refrigerator)
46 kitchen.add(refrigerator)
47 apple = M.new(type = 'f', name = 'Apple')
48 M.add_fact('edible', apple) # sagarra jangarria dela adierazi
49 refrigerator.add(apple)
50 tomato_plant = M.new(type = 'f', name = 'Tomato plant')
51 refrigerator.add(tomato_plant)
52 counter = M.new(type = 's', name = 'Counter')
53 kitchen.add(counter)
54 stove = M.new(type = 's', name = 'Stove')
55 kitchen.add(stove)
56 kitchen_island = M.new(type = 's', name = 'Kitchen island')
57 kitchen.add(kitchen_island)
58 note = M.new(type = 'o', name = 'Note')
59 kitchen_island.add(note)
60
61 # bathroom
62 toilet = M.new(type = 'c', name = 'Toilet')
63 M.add_fact('closed', toilet)
64 bathroom.add(toilet)
65 bath = M.new(type = 'c', name = 'Bath')
66 M.add_fact('closed', bath)
```

```
bathroom.add(bath)
soap = M.new(type = 'o', name = 'Soap bar')
bath.add(soap)
sink = M.new(type = 's', name = 'Sink')
bathroom.add(sink)
toothbrush = M.new(type = 'o', name = 'Toothbrush')
sink.add(toothbrush)

# backyard
chairs = M.new(type= 's', name = 'Set of chairs')
backyard.add(chairs)
bbq = M.new (type = 's', name = 'Bbq')
backyard.add(bbq)
patioTable = M.new(type = 's', name = 'Patio table')
backyard.add(patioTable)

# garden
lettuce = M.new(type = 'f', name = 'Lettuce')
M.add_fact('edible', lettuce)
garden.add(lettuce)
pepper = M.new(type = 'f', name = 'Bell pepper')
M.add_fact('edible', pepper)
garden.add(pepper)
chips = M.new(type = 'f', name = 'Half of a bag of chips')
M.add_fact('edible', chips)
garden.add(chips)
shovel = M.new(type = 'o', name = 'Shovel')
garden.add(shovel)

# living room
couch = M.new(type = 's', name = 'Couch')
livingRoom.add(couch)
milk = M.new(type = 'f', name = 'Milk')
M.add_fact('edible', milk)
couch.add(milk)
lowTable = M.new(type = 's', name = 'Low table')
livingRoom.add(lowTable)
remote = M.new(type = 'o', name = 'Remote')
lowTable.add(remote)
tv = M.new(type = 's', name = 'Tv')
livingRoom.add(tv)
```

```
108
109 M.set_player(bedroom) # jokalaria kokatu
110
111 # M.generate_distractors(n) n ausazko elementu sortu eta kokatu
112
113 # M.render() jokoa grafikoki ikusteko
114
115 quest = M.record_quest()
116
117 print( " > ".join(quest.commands))
118 print("\n" + quest.desc)
119
120 game = M.build()
121 game_name = "joko sinple"
122 game_file = M.compile(game_name)
```

GameMaker-ek jokoaren zenbait beste ezaugarri definitzeko aukera ematen du, esaterako, jokoaren helburua azken komandoa soilik izatea. Aukerak TextWorld-eko dokumentazioan[1] eskuragai daude.

[1] https://textworld.readthedocs.io/en/stable/textworld.generator.game.html?highlight=gameoptions#textworld.generator.game.GameOptions