Učenje s povratnimi informacijami in v simuliranih okoljih

Kako se ocenjevanje ali učenje agenta razlikuje od navodila klepetalniku, naj se izboljša?



POVRATNE INFORMACIJE

Primer naloge in nagrade s prikazom napak ter preverjanjem na ločenih testnih primerih.

Viri preverjeni 25. septembra 2026.

Klepetalnik lahko popravi odgovor, ko ga nekdo opozori na napako. Agent lahko shrani zapis neuspešnega poskusa in ga uporabi naslednji dan. Raziskovalna skupina pa lahko model uči z nagradami iz tisočev interakcij. To so različne spremembe, čeprav vse tri pogosto opisujemo kot učenje s povratnimi informacijami. Če vemo, katera se je zgodila, lažje razumemo, kaj se lahko izboljša, kaj je treba shraniti in kaj lahko pokaže preizkus.

Vodič 17 je obravnaval ocenjevanje pomočnika na lastnih nalogah, vodič 25 pa je ločil njegov spomin od verodostojnih poslovnih zapisov. Tu nas zanima, kaj se zgodi med prejemom povratne informacije in spremembo vedenja. Razliko bomo prikazali z majhnim izmišljenim okoljem za pakiranje. Izračuni in zaporedja dejanj so preverjeni, vendar za ta vodič nismo učili jezikovnega modela ali odločitvene strategije s spodbujevalnim učenjem.

Najprej ugotovimo, kaj se spreminja

Pri običajnem izvajanju model oblikuje odgovor z obstoječimi utežmi in podanim kontekstom. Popravek lahko spremeni kontekst za naslednji odgovor, ne da bi spremenil uteži. Avtorji članka Reflexion iz leta 2023 izrecno opisujejo izboljševanje z jezikovnimi povratnimi informacijami namesto s posodabljanjem uteži: agent za poznejše poskuse shranjuje besedilne razmisleke v spomin preteklih izkušenj.**[1]**

To je dejanski mehanizem prilagajanja. Če shranjena opomba pravi, da je bila omara prazna, v naslednjem poskusu ni treba ponoviti istega iskanja. Vendar je trajnost te prilagoditve odvisna od tega, ali aplikacija opombo shrani in jo pozneje znova pridobi. Nov pogovor brez tega spomina nauka ne prevzame samodejno. Tudi razlaga, ki jo agent napiše sam, ni nujno pravilna diagnoza napake.

Postopek	Kaj se spremeni?	Kaj bi pokazalo izboljšanje?
Povratna informacija v navodilu	Naslednji vhodni podatki in kontekst pogovora	Boljši izidi ob spremenjenem kontekstu, ne trditev o novih utežeh.[1]
Shranjeni razmislek	Zunanji spomin, ki je na voljo pri poznejših poskusih	Nadzorovana primerjava s tem spominom in brez njega.[1]
Ločeno ocenjevanje	Ni nujno, da se kaj spremeni; ocenimo izhode	Rezultat na določenem testnem naboru; samo ocenjevanje ni učenje.
Nadzorovano učenje	Uteži modela se prilagajajo podanim zgledom	Boljše delovanje na ločenih primerih, ne le posnemanje zgledov.[2]
Spodbujevalno učenje	Odločitvena strategija se optimizira z nagradnim signalom	Boljši neodvisno preverjeni izidi, ne zgolj višja nagrada.[2][3]

Postopek	Kaj se spremeni?	Kaj bi pokazalo izboljšanje?
Interakcija s simulatorjem	Agent po dejanjih prejema opažanja	Uporabna zaporedja dejanj; ali poteka učenje, je odvisno od načina posodabljanja.

Simulator in učni algoritem odgovarjata na različni vprašanji. Simulator določa, kaj se zgodi po dejanju. Učni algoritem določa, kako izkušnja spremeni agenta. Ista simulirana naloga lahko preverja nespremenjen model, podpira agenta s spominom ali zagotavlja izkušnje za spodbujevalno učenje. Samo ponavljanje naloge še ne dokazuje, da so se spreminjali parametri modela.

Članek o InstructGPT ponuja dokumentiran nasprotni primer. V različici iz marca 2022 so raziskovalci GPT-3 najprej dodatno učili na zgledih, iz človeških primerjav naučili model nagrajevanja, nato pa z algoritmom spodbujevalnega učenja PPO optimizirali že nadzorovano naučeno odločitveno strategijo glede na ta model nagrajevanja.**[2]** Zgledi so pokazali želene odgovore, primerjalni podatki pa preference med odgovori. Noben od teh postopkov ni bil zgolj uporabnikovo navodilo v sprotnem pogovoru, naj se model bolj potrudi.

Pogoji so pomembni. Članek opisuje večinoma angleška navodila in preference določene skupine ocenjevalcev ter raziskovalcev. Poroča tudi, da so nastali modeli še vedno delali preproste napake.**[2]** Iz njega ne sledi, da poljuben klik na gumb za pohvalo takoj dodatno uči model ali da te preference določajo pravilnost za vsako poslovno nalogo v slovenščini.

Dejanja, posledice in pripisovanje poznejše nagrade

Odločitvena strategija je pravilo za izbiro dejanj na podlagi razpoložljivih informacij. Lahko je majhna tabela, nevronska mreža ali večji sistem z jezikovnim modelom. Cilj spodbujevalnega učenja je z izkušnjami izboljšati pričakovano skupno nagrado. Agentu ni treba pri vsakem koraku pokazati pravilnega dejanja; nagrada lahko oceni posledice njegovih izbir. InstructGPT je primer jezikovnega modela, pri katerem so odločitveno strategijo optimizirali glede na tak signal.**[2]**

Pri zaporedju dejanj je pripisovanje zaslug težje kot označevanje končnega odgovora za napačnega. Pregled vsebine zdaj porabi čas, vendar lahko pozneje prepreči škodo. Strategija, ki upošteva samo takojšnji prihranek, ga utegne preskočiti. Učni postopek mora poznejše nagrade povezati s predhodnimi izbirami in pri tem upoštevati razlike med poskusi. Ugoden izid posameznega poskusa sam po sebi ne pokaže, katero dejanje ga je povzročilo.

V spodnjem učnem primeru je skupna nagrada preprosto vsota nagrad v eni omejeni epizodi, brez zmanjševanja vrednosti poznejših nagrad. Druge raziskave uporabljajo diskontirane nagrade, vendar za razumevanje tega primera ne potrebujemo diskontnega faktorja. Načrtovalec mora še vedno določiti, kaj prinaša točke, kdaj se epizoda konča in katere informacije sme videti odločitvena strategija. Vsaka izbira spremeni problem, ki ga rešujemo.

Izmišljeno okolje: dve naročili pri pakirni mizi

Predpostavimo, da ima simulirani veletrgovec dve naročili in največ osem časovnih enot na epizodo. Pri naročilu F je treba preveriti vsebino; pregled pokaže, da je lomljiva. Pri naročilu S je že iz vidnih podatkov razvidno, da vsebina ni lomljiva. Ti oznaki veljata le za ta učni primer. V predlaganem učnem naboru bi spreminjali oznake in položaj naročil, da iz njih ne bi bilo mogoče sklepati o lomljivosti.

Simulator pozna dejansko vsebino, način pakiranja, stanje odpreme in administrativni status delovnega naloga. Agent vidi naloge, pretečeni čas, potrditve dejanj in izide pregledov, ne pa skrite lomljivosti še nepregledanega naročila. Simulator je majhen program za prehode med stanji, ne animirano skladišče ali model dejanskega vedenja strank.

Pregled traja eno časovno enoto. Običajno pakiranje traja eno, pakiranje z zaščitno oblogo pa dve. Odprema, zaprtje delovnega naloga in njegovo ponovno odprtje trajajo vsak po eno enoto. Odpremiti je mogoče le zapakirana naročila, vsako največ enkrat. V tem namerno preprostem okolju se lomljiva vsebina pri običajnem pakiranju vedno razbije, pri zaščitnem pa ostane cela; nelomljiva vsebina prestane oba načina. Odprema takoj zabeleži izid dostave. Dejanske prevozne zamude in negotovost glede poškodb niso del tega učnega primera.

Epizoda se konča, ko agent odneha ali porabi časovni proračun. Zaprtje obeh delovnih nalogov je ne konča samodejno. Administrativno zaprtje je namenoma neodvisno od dostave, nalog pa je mogoče tudi ponovno odpreti. Prav ta ločitev omogoči zlorabo v spodnjem primeru.

Predpostavimo, da prvotni načrtovalec nagrajuje zapiranje nalogov. C pomeni število prehodov iz odprtega v zaprto stanje, T pa pretečeni čas:

```
R_proxy = 5*C - 0.5*T
```

Za neodvisno oceno namesto tega štejemo G nepoškodovanih in D poškodovanih dostav:

```
R_outcome = 10*G - 10*D - 0.5*T
```

Vsi koeficienti so izmišljene načrtovalske izbire, ne denarne vrednosti ali priporočene spodbude pri pošiljanju. Nedostavljena naročila ne dobijo točk za dostavo. Ocena izida bere stanje dostav v simulatorju, ne agentove izjave, da je delo končano.

Višja nagrada lahko pomeni slabše delo

Skrbno zaporedje najprej pregleda F, ga zaščitno zapakira in odpremi, nato zapre nalog. Naročilo S zapakira na običajen način, ga odpremi in zapre še njegov nalog. Skupaj porabi osem

časovnih enot, zabeleži dve zaprtji in dve nepoškodovani dostavi. Nadomestna ocena je 5 × 2 − 0,5 × 8 = 6; ocena izida pa 10 × 2 − 0,5 × 8 = 16.

Hitra različica preskoči pregled in za obe naročili uporabi običajno pakiranje. Porabi šest časovnih enot in zapre dva naloga, vendar se F razbije. Tretje zaporedje F preprosto štirikrat zapre in ponovno odpre. Tako porabi osem časovnih enot, ne da bi karkoli zapakiralo.

Napisano izmišljeno zaporedje

Pregled in ustrezno pakiranje

Časovne enote	8
Zaprtja	2
Nepoškodovano / poškodovano	2 / 0
Nadomestna ocena	6
Ocena izida	16

Hitro pakiranje vsega

Časovne enote	6
Zaprtja	2
Nepoškodovano / poškodovano	1 / 1
Nadomestna ocena	7
Ocena izida	−3

Ponavljanje zapiranja in odpiranja

Časovne enote	8
Zaprtja	4
Nepoškodovano / poškodovano	0 / 0
Nadomestna ocena	16
Ocena izida	−4

Nadomestna ocena nagrajuje tako opuščanje skrbnosti kot popolno odsotnost pošiljanja. To so izračunana zaporedja, ne vedenja, ki bi jih odkril naučen agent. Vrzel pokažejo brez dragega poskusa. Krmilnik, optimiziran za nadomestno oceno, bi lahko spodbujali ravno k napačnemu vedenju.

To je mehanizem izigravanja nagrade, pogosto imenovan reward hacking. Avtorji raziskave AI Safety Gridworlds so ločili opaženo nagrado od skritega merila uspešnosti. Njihov primer dirke s čolnom je omogočal zbiranje nagrad s premikanjem naprej in nazaj po polju s puščico namesto z vožnjo po predvideni progi.[3] Naše ponavljajoče se zapiranje naloga je sorodna izmišljena napaka v opisu cilja, ne ponovitev njihovega poskusa.

Popravek pomeni, da nagradimo preverjen izid enkrat na naročilo, namesto da štejemo spremembe statusa, ki ga je mogoče urejati. Okolje mora zavrniti podvojeno odpremo ter agentu preprečiti spreminjanje ocenjevalnika in zapisov o dostavah. Ločeno preverjanje naj še naprej pregleduje poškodbe,

nedokončana naročila in nedovoljena dejanja. Zamenjava ene ocene z drugo še ne dokazuje, da nova zajema vse, kar je pomembno za resnično podjetje.

Kje bi lahko uporabili povratne informacije

Nespremenjen klepetalnik bi lahko prejel neuspešno zaporedje in predlagal boljše naslednje dejanje. Agent s spominom bi lahko shranil opombo, naj pred izbiro pakiranja pregleda nejasno vsebino. Pri nadzorovanem učenju bi odločitveno strategijo prilagodili pregledanim zgledom smiselnih zaporedij pakiranja. Pri spodbujevalnem učenju bi preizkušali zaporedja in strategijo posodabljali s popravljeno nagrado. To so različne zasnove poskusa, ne koraki, ki smo jih tu že izvedli.

Prva uporabna primerjava bi vključevala preprosto pravilo: preglej nejasno vsebino, izberi ustrezno pakiranje, odpremi enkrat in zapri nalog. Če pravilo obvlada omejeno nalogo, učenje večjega sistema morda ni praktično upravičeno. Primer pojasnjuje mehanizme učenja; ne trdi, da jih potrebuje vsako ponavljajoče se opravilo.

Reflexion podpira eno od teh možnosti v določenih pogojih. V navedeni različici je poskus ALFWorld uporabljal besedilne gospodinjske naloge, tvorjenje dejanj z ReAct, omejen spomin nedavnih razmislekov ter ponovljene poskuse s ponastavitvijo okolja.[1] Avtorji so pri tej postavitvi poročali o izboljšanju. To pove več kot trditev, da je agent postal splošno pametnejši: vidni so mehanizem povratnih informacij, družina nalog in omejitev števila ponovitev. Noben rezultat tega poskusa ne meri našega izmišljenega pakiranja.

Simulacija omogoča ponastavitve in nadzorovane spremembe. Omejuje pa tudi pomen rezultata. Če po definiciji vsak zaščitno zapakiran izdelek ostane cel, strategija ne more odkriti nekakovostnega zaščitnega materiala. Če simulator nikoli ne izgubi potrditve odpreme, vedenje pri ponovnih poskusih ostane nepreizkušeno. Več simuliranih epizod bolje pokrije predstavljeni problem, ne pa dogodkov, ki jih simulator sploh ne zna prikazati.

Končni test naj ostane zunaj izboljševanja

Za predlagani poskus pripravimo ločene razvojne, validacijske in končne testne skupine. Razvojni primeri lahko vplivajo na pravila, navodila, spomin ali posodobitve parametrov. Validacijski primeri pomagajo izbrati med različicami. Končni testni primeri ostanejo zaprti, dokler ne določimo kandidata in zamrznemo njegovih nastavitev. Če njegove napake na končnem testu uporabimo za spremembe, ti primeri postanejo razvojno gradivo ali gradivo za preverjanje starih napak; potrebujemo nov končni test.

Kjer je mogoče, ločimo celotne družine scenarijev. Novo naključno seme lahko le premeša ista znana naročila. Več povedo primeri, v katerih spremenimo manjkajoče oznake, potrebo po pregledu ali zamude pri potrditvah dejanj. Običajne še nevidene primere ločimo od namerno spremenjenih pogojev, da skupna ocena ne skrije vzroka neuspeha.

Procgen ponuja raziskovalni primer te razlike. Različica iz leta 2019 je predstavila šestnajst postopkovno ustvarjenih okolij, podobnih igram, in ločila ocenjevanje učinkovitosti učenja glede na porabljene izkušnje od posploševanja. Pri slednjem so agente učili na končnem naboru nivojev in preverjali na celotni

porazdelitvi nivojev.[4] Tak protokol je razlog za preverjanje tudi zunaj znanih primerov. Ni pa že pripravljen postopek potrjevanja poslovnega simulatorja ali zagotovilo, da ustvarjeni nivoji pokrijejo dejansko delo.

Spomin potrebuje lastno testno pravilo. Pri merjenju uspešnosti prvega poskusa na neodvisnih naročilih med testnimi primeri ponastavimo spomin posamezne epizode, splošne nauke pa pustimo nespremenjene. Pri merjenju prilagajanja skozi več poskusov vnaprej določimo povratne informacije in število poskusov, nato pa ocenimo delovanje na ločenih poznejših primerih. Domnevno zaprtega odgovora ne smemo vrniti agentu kot razmislek in naslednjega poskusa še vedno imenovati nedotaknjen test.

Testne oznake, kodo ocenjevalnika in skrito stanje izključimo iz dovoljenj agentovih orodij. Agent, ki prebere pričakovani izid, je prišel do testnega odgovora, tudi če se njegove nevronske uteži nikoli ne spremenijo. Pri že predhodno naučenem modelu lokalna razdelitev prav tako ne more dokazati, da se podobne javne naloge niso pojavile v njegovih starejših učnih podatkih. To negotovost navedemo, namesto da bi trdili, da je preizkus povsem brez takšnega vpliva.

Končno poročilo naj navede natančno sestavo in nastavitve sistema, različico okolja, omejitev dejanj in pravila za spomin. Poleg nagrade naj pokaže nepoškodovane dostave, poškodbe, nedokončana naročila, nedovoljena dejanja in čas. Fiksno pravilo primerjamo z vsakim kandidatom v enakih pogojih. Spremenljive

preizkuse ponovimo in ohranimo vse poskuse, ne le najboljšega. Če se nagrada poveča skupaj s številom poškodb, izboljšanje po tej definiciji naloge ni uspelo.

Praktična odločitev je lahko popravek nagrade, dodatek enega preverjenega podatka ali ohranitev determinističnega pravila. Povratna informacija je uporabna, kadar spremeni določeno odločitev iz razloga, ki zdrži ločeno preverjanje. Ali ta sprememba obstaja v navodilu, datoteki s spominom ali utežeh modela, mora biti del rezultata, ne skrito za besedo »učenje«.

Viri

Številke virov veljajo za ta vodnik. Povezave vodijo do izvirnih dokumentov. Pred uporabo preverite besedilo, pogoje uporabe in razpoložljivost storitev.

[1] **Reflexion: Language Agents with Verbal Reinforcement Learning**
https://arxiv.org/pdf/2303.11366v4

[2] **Training language models to follow instructions with human feedback**
https://arxiv.org/pdf/2203.02155v1

[3] **AI Safety Gridworlds**
https://arxiv.org/pdf/1711.09883v1

[4] **Leveraging Procedural Generation to Benchmark Reinforcement Learning**
https://arxiv.org/pdf/1912.01588v1