Introduzione (2/3 pg)

Qui pensavo di mettere una sorta di riassunto di questo documento. In ogni caso:

- Contesto: AI, sistemi generativi (LLM)
- Problema: Qual è il livello dell'AI nel pianificare e raggiungere un obiettivo. Ne sono in grado?
- Stato dell'arte: PDDL (con esempio al progetto fatto nel corso di AsA), Reinforcement Learning (ho trovato un paper che tratta un problema simile ma pensavo di citarlo al meglio nella parte "state of the art" nel primo capitolo), inoltre ho trovato un paio di paper che trattano il planning negli LLM
- Problema specifico: Gli LLMs "senza sovrastruttura" sono in grado di risolvere un problema di logistica?
- Obiettivo: Verificare se ne sono in grado ed eventualmente fino a che punto
- Risultati attesi: Trovare limiti e potenzialità di un LLM in questo contesto
- Struttura: da definire bene alla fine

Capitolo 1 - Background

Qui pensavo di mettere le informazioni necessarie alla comprensione della tesi, quindi:

- LLMs: prima un'introduzione all'NLP fino ai Transformer. Mentre nel dettaglio vorrei sottolineare:
    - Attention mechanism: per spiegare poi il problema dell'attenzione sui caratteri della mappa nel prompt
    - Few shot learners: per spiegare poi perché funziona meglio se uso tutto lo storico della chat
- Incertezza negli LLM: spiegare il discorso delle allucinazioni, cosa sono e poi i vari modi che la letteratura ha usato per studiare l'incertezza
    - KnowNo framework: che è l'approccio che ho usato
    - Altri approcci (i paper trovati all'inizio del progetto): per dare una visione completa dell'argomento
- Agenti/Agenti BDI: per dare un'idea generale di cosa sia un agente e come funzioni
- Stato dell'arte:
    - PDDL: pro (explainable e funzionante se il l'albero non è enorme) e contro (time intensive, non si adatta "in automatico", in un problema anche moderato è quasi impossibile usarlo per il real time)
    - Reinforcement Learning: pro (efficace, si può adattare all'ambiente che cambia) e contro (potrebbe trovare un local minima nella reward function, no explainability, costa trainare)
    - Agenti LLM (tipo o1): hot topic attualmente, cosa sono e come si differenziano dal mio approccio (quindi il discorso di Chain of Thought e come "pianificano")

Capitolo 2 - Experiment setting

Qui pensavo di descrivere come ho testato tutto: l'ambiente (deliveroo), gli strumenti usati (API OpenAI), i modelli usati, i prompt usati, ecc.

- Definizione del problema: Questo è un approccio generativo, non c'è ragionamento, non c'è pianificazione, non c'è path finding. L'idea è di testare le potenzialità di un agente in un ambiente che cambia molto velocemente senza sapere a priori "cosa vedrà": questa cosa è emulata non parsando le informazioni dal server e usando informazioni grezze della mappa nel prompt, in modo che se cambia la struttura delle risposte di Deliveroo, il sistema continua a funzionare.
- Deliveroo.js: spiegare l'ambiente del test
- Modelli GPT: cosa sono, come funzionano

    - perché GPT: l'ho scelto perché è il più conosciuto, molto potente e l'API ha i parametri necessari a far funzionare il KnowNo framework
    - perché non LLAMA o altri modelli open source/open weight -> non è triviale l'implementazione del corrispettivo dei parametri logit bias e logprobs (richiederebbe di non usarli tramite framework ma di ricostruire il modello con Pytorch, caricare i pesi e modificare pesantemente la struttura), inoltre hanno una peggior comprensione del prompt (se dico "rispondi con la lettera dell'azione" la risposta non è "U" ma "Certo! L'azione è U" che renderebbe quindi impossibile andare a fare tutti i calcoli sulle logprob)
    - API OpenAI: come l'ho usata e focus sui parametri "importanti" (per esempio la temperatura settata a 0 per riproducibilità e per ottenere le performance "grezze" del LLM)

Capitolo 3 - Agent development

Qui pensavo di descrivere l'evoluzione nello sviluppo, sottolineando i problemi e il perché ho cambiato ogni approccio.

- Primo approccio: molto try-and-error. Ho iniziato parsando le informazioni dal server e implementando 7 parametri uno dopo l'altro controllando nei test dove l'agente aveva problemi (tipo loopare o scegliere azioni impossibili) -> abbandonato perché lo sviluppo stava andando nella direzione sbagliata, introducendo troppi aiuti non stavo più studiando le reali capacità del LLM
- Secondo approccio: stateless full raw. Ho messo solo le informazioni grezze (ricevute dal server deliveroo) nel prompt lasciando che il LLM capisse da solo la mappa, l'obiettivo e tutto, ma i prompt erano sbagliati e i risultati erano ovviamente pessimi
- Agente finale: nuovi prompt seguendo la letteratura, aggiungendo anche l'approccio stateful (con lo storico della chat), risultati ancora non positivissimi ma più solidi, studiabili e riproducibili e ho iniziato a raccogliere dati per comprendere più a fondo le limitazioni
- Extra: "qual è, tra le celle vicine, la più vicina all'obiettivo?" e "come ci arrivo?" -> per vedere se il LLM è in grado di capire l'orientamento della mappa: qui ci sono state varie problematiche che voglio discutere meglio nel capitolo "Results discussion". In linea di massima qui vorrei solo esporre l'implementazione e la struttura di questo approccio.

Capitolo 4 - How data has been collected

Qui pensavo di descrivere come ho raccolto i dati, i test fatti, i risultati raccolti, cosa sono le heatmap create, ecc.

- Attention visualizer in BERT: per mostrare perché la mappa grezza potrebbe essere un po' difficile da capire per il LLM a causa di tutti i caratteri come {, }, ecc. Cito anche i test fatti codificando la mappa (base64) per ridurre il numero di caratteri che però non ha ridotto il numero di token
- Prompt usati: mostro i prompt e spiego alcune scelte, sottolineando anche alcune decisioni, come ad esempio mettere la parte che cambia del prompt alla fine del prompt (la posizione dell'agente che viene aggiornata ad ogni query) in modo che potessi sfruttare i prezzi ridotti dell'API grazie al caching; discuto brevemente l'effetto di aggiungere il risultato di un'azione nel prompt (che sarebbe "barare" perché il LLM risponderebbe basandosi sulle sue capacità "matematiche" e non sulla mappa effettiva)
- Scelte nella creazione del prompt, in accordo con i paper: dare un ruolo al LLM, dove mettere l'obiettivo all'interno del prompt, KnowNo per la parte "dammi solo l'azione". Il fatto che stateful "funzioni" è comunque un indicatore che la struttura del prompt sia corretta (per il discorso di few shot learners)
- Creazione delle heatmap: come si fanno, come si leggono, come si usano, perché sono utili

Results discussion

- Stateless: mostro cosa le heatmap hanno in comune se grandi o piccole (tipo l'avere problemi nelle celle orizzontali e verticali) cercando di dare una spiegazione (solo una azione è corretta, più probabile Pickup e Putdown nei pressi della cella goal...); sottolineo che le "stesse celle" in mappe diverse hanno diversi valori (per esempio le celle attorno al goal nella mappa 5x5 hanno valori diversi rispetto alle celle attorno al goal nella mappa 7x7)
- Stateful: visualizzazione del path fatto dall'agente, mostro che con l'aumentare della dimensione della mappa, l'agente ha più problemi a raggiungere l'obiettivo e a causa della limitazione del numero di token nel contesto (e il prompt è abbastanza grande perché contiene la mappa) sprecare anche solo un paio di azioni andando in loop satura la lunghezza del prompt; valuto alcuni esempi confrontando il percorso preso dall'agente con il percorso ottimale (ho creato delle visualizzazioni ad hoc per questo)
- Stateless & Stateful: stateful funziona meglio a causa di "LLMs are few-shot learners" e lo storico della chat con azione-effetto_nella_mappa aiuta il LLM a capire la mappa e l'obiettivo; inoltre, come vediamo nella stateless, in certi casi se l'agente finisce in porzioni di mappa particolari, va in loop e da lì non esce più. Le zone problematiche, nonstante in percentuale non cambino molto rispetto alla dimensione della mappa, in numeri reali sono porzioni di mappa anche sostanziose (quindi se per esempio in una mappa 3x3 abbiamo una cella problematica e in una mappa 21x21 ne abbiamo 49, in percentuale sono lo stesso numero, ma se nel primo caso l'agente finisce nella cella problematica molto facilmente se ne va e arriva al goal, nel secondo caso l'agente non ne esce più)
- Extra: cella che più mi avvicina al goal: prima di tutto vediamo un bias nell'orientamento della mappa come si può vedere dalle heatmap fatte appositamente (la cella [0,0] viene considerata come quella "da programmatore" quindi in alto a sinistra e non in basso a sinistra come nel piano cartesiano). Il problema qui è che il LLM è propenso a dare una tile sbagliata come risposta e quindi, qualsiasi azione poi per raggiungere quella cella specifica, mi allontana dal goal.
- Differenza tra i vari modelli testati (3.5, 4o, 4o-mini) -> effettivamente più un modello è "nuovo e potente" meglio va, indipendentemente dal numero di parametri -> vorrei inserire alcuni benchmark "standard" per mostrare le differenze

Conclusions (2 pg)

L'obiettivo era applicare un LLM ad un problema di logistica senza sovrastrutture. I risultati hanno dimostrato che dipende molto dalla dimensione della mappa e che, nonostante a livello relativo, la percentuale di celle problematiche sia bassa, in numeri reali questo si traduce in porzioni di mappa in cui, se l'agente ci va, non ne esce più.

I risultati sono stati interessanti, ma ci sono delle limitazioni. La più grande è che l'agente non ha nessuna capacità di ragionamento, quindi non è in grado di fare path finding, di fare planning, di fare reasoning. Questo è un problema che si può risolvere con PDDL o con RL, ma l'idea era di vedere se un LLM "nudo" fosse in grado di risolvere un problema di questo tipo. E la risposta è sì, ma con delle limitazioni.

Limitazioni: il prompt è stato fatto seguendo la letteratura ma potrebbe non essere il migliore. Il funzionamento del logit bias di OpenAI cambia nel tempo (a fine 2023 forzava l'output solo sui token selezionati, ora aumenta di molto la loro probabilità ma è comunque possibile escano token non voluti; potrebbe cambiare di nuovo in futuro) e questo potrebbe rendere la riproducibilità dei risultati difficile. Il numero di token è limitato e questo potrebbe essere un problema in mappe più grandi, soprattutto se si sbagliano azioni. Ci possono essere dei bias nelle risposte come mostrato dove i risultati sono stati discussi.

Sviluppi futuri:

- usare un modello più potente in quanto si vede che effettivamente modelli migliori danno risultati migliori
- implementazione di RAG per aiutare nella decisione (magari inserendo delle preferenze di percorso, specifiche informazioni in base ad una categorizzazione delle parcel, ecc.)
- si potrebbe effettivamente mettere mano a dei modelli open e testarli
- provare con modelli multimodali e mettere uno screenshot dell'environment nel prompt.