Exploring the Use of LLMs for Agent Planning: Strengths and Weaknesses

Introduzione (2/3 pg)

In questo capitolo pensavo di inserire una forma riassunta ma discorsiva di questo documento. In particolare, questi sono i punti che vorrei trattare:

- **Contesto**: AI, sistemi generativi (LLM)
- **Problema**: verifica della capacità degli LLM di pianificare e raggiungere obiettivi in autonomia senza sovrastrutture
- **Stato dell'arte**: analisi di approcci consolidati come PDDL (con esempio al progetto fatto nel corso di AsA), Reinforcement Learning (ho trovato un paper che tratta un problema simile che vorrei trattare più a fondo nella parte "state of the art" del primo capitolo), inoltre ho trovato un paio di paper che trattano il planning negli LLM che vorrei citare
- **Problema specifico**: valutazione della capacità di un LLM, privo di sovrastrutture esterne, di risolvere problemi di logistica in ambienti dinamici
- **Obiettivo e Risultati attesi**: identificazione delle principali limitazioni e delle aree di forza degli LLM nell'ambito della pianificazione.

La parte della **struttura** del documento sarà definita al termine della stesura della tesi.

Capitolo 1 - Background

Questo capitolo fornisce il contesto teorico necessario alla comprensione del lavoro, trattando i seguenti argomenti:

- **LLMs**: introduzione alle tecnologie di NLP con particolare riferimento ai Transformer. Verranno approfonditi i seguenti aspetti:
    - *Attention mechanism*: per spiegare il problema dell'attenzione sui caratteri "speciali" nella definzione della mappa inserita nel prompt
    - *Few shot learners*: per spiegare le migliorie dovute all'avere lo storico della chat nel prompt
- **Incertezza negli LLM**: introdurre le allucinazioni per poi illustrare i vari modi che la letteratura ha usato per studiare l'incertezza negli LLM:
    - *KnowNo framework*: che è l'approccio che ho usato
    - *Altri approcci* (i paper trovati quando ho iniziato a lavorare 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, efficienza in ambienti limitati) e contro (time intensive, non c'è adattamento, in un problema di dimensioni moderate è quasi impossibile usarlo in real time)
    - *Reinforcement Learning*: pro (efficace, si può adattare all'ambiente che cambia) e contro (convergenza ad un local minima nella reward function, no explainability, costoso da allenare)
    - *Agenti LLM (es. o1 di OpenAI)*: introduzione a temi emergenti come il Chain of Thought, sottolineando come viene eseguito il reasoning e cosa differisce dal mio approccio

Capitolo 2 - Experiment setting

Questo capitolo descrive il framework sperimentale utilizzato.

- **Definizione del problema**: 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 cambiasse la struttura delle risposte di Deliveroo.js, il sistema continuerebbe a funzionare
- **Deliveroo.js**: spiegare l'ambiente del test
- **Modelli GPT**: cosa sono, come funzionano
    - *perché GPT*: molto potente, molto conosciuto 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` (i framework non li supportano, sarebbe necessario ricostruire il modello con Pytorch, caricare i pesi e modificare pesantemente la struttura), inoltre hanno una peggior comprensione del prompt (es. alla richiesa "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 impostata a 0 per riproducibilità e per ottenere le performance "grezze" del LLM)

Capitolo 3 - Agent development

Il capitolo illustra il processo iterativo di sviluppo dell'agente, evidenziando problemi incontrati e soluzioni adottate.

- **Primo approccio**: molto try-and-error. Ho iniziato parsando le informazioni dal server, implementando poi 7 parametri di aiuto in base ai problemi che si sollevavano durante i test; abbandonato perché lo sviluppo stava andando nella direzione sbagliata, introducendo troppi aiuti non era più uno studio delle reali capacità del LLM
- **Secondo approccio**: stateless full raw. Solo le informazioni grezze (ricevute dal server Deliveroo) nel prompt lasciando che il LLM capisse da solo mappa e obiettivoo, ma i prompt erano sbagliati e i risultati erano 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;
- **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 vorrei solo esporre l'implementazione e la struttura di questo approccio.

Capitolo 4 - How data has been collected

In questa sezione vengono descritti i metodi utilizzati per raccogliere e analizzare i dati sperimentali.

- **Attention visualizer in BERT**: per mostrare come la mappa grezza potrebbe essere difficile da capire per un 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; usato BERT per semplicità di implementazione, ma il concetto è lo stesso

- **Prompt usati**: mostro i prompt e spiego alcune scelte, sottolineando anche alcune decisioni:
  - mettere dati variabili alla fine del prompt per sfruttare i prezzi ridotti dell'API grazie al caching
  - perché non aggiungere l'effetto dell'azione: sarebbe "barare" in quanto 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, in che parte inserire l'obiettivo, paper KnowNo per avere solo una azione come risposta.
  - il fatto che stateful "funzioni" è comunque un indicatore che la struttura del prompt sia corretta
- **Creazione delle heatmap**: come si fanno, come si leggono, come si usano, perché sono utili

Results discussion

- **Stateless**: cosa le heatmap hanno in comune indipendentemente dalla dimensione (problemi nelle celle orizzontali e verticali rispetto al goal) cercando di dare una spiegazione (solo una azione è corretta, inoltre più probabilità di Pickup e Putdown nei pressi della cella goal, ecc.); le stesse celle in mappe diverse hanno diversi valori (es: 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; con l'aumentare della dimensione della mappa, l'agente fa più difficoltà a raggiungere l'obiettivo e, a causa della limitazione del numero di token nel contesto, sprecare anche solo un paio di azioni satura la lunghezza del prompt; valuto alcuni esempi confrontando il percorso preso dall'agente con il percorso ottimale (ho creato delle visualizzazioni ad hoc)
- **Stateless & Stateful**: stateful funziona meglio per "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, 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 ne esce 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 facilmente da una tile sbagliata come risposta e quindi, qualsiasi azione poi per raggiungere quella cella mi allontana dal goal, inoltre è difficile applicare il calcolo dell'incertezza avendo due passaggi per arrivare all'azione
- **Differenza tra i vari modelli testati (3.5, 4o, 4o-mini)**: più un modello è "nuovo e potente" meglio va, indipendentemente dal numero di parametri -> vorrei inserire anche alcuni benchmark "standard" per mostrare le differenze

Conclusions (2 pg)

L'obiettivo era applicare un LLM ad un problema di logistica senza sovrastrutture aggiuntive. I risultati ottenuti evidenziano una forte dipendenza dalla dimensione della mappa: sebbene in termini relativi la percentuale di celle problematiche sia ridotta, in valori assoluti tali aree possono rappresentare porzioni significative della mappa. Quando l'agente si trova in queste aree, spesso non riesce a uscirne, compromettendo il raggiungimento dell'obiettivo.

I risultati sono stati certamente interessanti, ma presentano alcune limitazioni significative. La principale è l'incapacità dell'agente di eseguire operazioni di ragionamento avanzato, quali il path finding, la pianificazione strategica e il reasoning. Tali capacità potrebbero essere implementate mediante l'uso parallelo di approcci tradizionali come il PDDL. Tuttavia, l'obiettivo di questo studio era valutare la capacità di un LLM “nudo” di affrontare autonomamente un problema di questo tipo. La risposta è positiva, ma con limiti evidenti.

Limitazioni: il prompt, sebbene costruito in accordo con la letteratura, potrebbe non essere ottimale. Inoltre, il comportamento del logit bias delle API OpenAI, che inizialmente (fine 2023) forzava l'output solo sui token selezionati, è soggetto a variazioni. Attualmente, questa funzione incrementa la probabilità dei token desiderati, ma non esclude completamente altre risposte, e potrebbe subire ulteriori modifiche nel tempo, con potenziali implicazioni per la riproducibilità dei risultati. Un'altra limitazione riguarda il numero di token disponibili nel contesto, che diventa critico in mappe di grandi dimensioni, soprattutto in presenza di errori nelle azioni compiute dall'agente. Infine, sono stati evidenziati alcuni bias nelle risposte, discussi nei risultati.

Sviluppi futuri:

- utilizzare modelli più avanzati, dato che i risultati migliorano con l'incremento delle capacità del modello
- integrazione di RAG per supportare le decisioni, ad esempio inserendo preferenze di percorso o informazioni specifiche basate su una categorizzazione delle parcelle
- esplorare modelli open-source e testarli, nonostante le difficoltà tecniche legate alla loro implementazione
- sperimentare con modelli multimodali, includendo uno screenshot dell’ambiente all’interno del prompt per migliorare la comprensione del contesto
- sicuramente me ne ricorderò di più durante la stesura della tesi