Introduction (2/3 pages)

Exploring uncertainty in raw LLMs as agents in a logistics problem

> context, problem, state-of-the-art, specific problem of the thesis, objective/objectives, expected results, structure

context: AI, Generative systems problem: What is the level of AI in planning and reaching a goal. Are LLM capable? state-of-the-art: LLM applied to similar problems (paper with "EXPLORING AND BENCHMARKING PLANNING CAPABILITIES OF LARGE LANGUAGE MODELS", "PDDL PLANNING WITH PRE-TRAINED LARGE LANGUAGE MODELS") specific problem of the thesis: use an LLM "naked" to solve a problem of logistics objective/objectives: check the effectiveness of an LLM of a typical AI problem (logistics) expected results: a generative system without any reasoning capabilities is able to solve a problem of this type? And to what extent?

Everyone says LLM are incredible for everything. My thesis: let's try to apply them to the delivery problem. A very simple problem, a classic problem for AI, that is composed by:

- a planing problem
- a reasoning problem
- ...

This can be faced in a classic way with PDDL (time intensive), RL. We want to address this with LLMs.

Chapter 1 - Background

> state-of-the-art, background

- LLMs: the evolution of NLP that brought to LLMs
  - Focus on Attention mechanism: because to explain the problem of attention over the characters of the map in the prompt
  - Few shot learners: to explain why it works better if I keep the chat history
- Uncertainty in LLMs: what is it and the different ways to compute it
  - KnowNo framework: that's the approach I'm going to use
  - Other approaches: to give a complete view of the topic
- Agents/BDI Agents: to give a general idea of what an agent is and how it works
- SotA: PDDL
  - pros (effective) and cons (time intensive)
- SotA: Reinforcement Learning
  - pros (effective) and cons (local minima)
- SotA: LLM agents: hot topic currently, what it is and how it differs from my approach

Chapter 2 - Experiment setting

Describe where the experiment has been conducted, the environment, the tools used, the models used, the prompts used, ecc.

- Problem definition: This is a generative approach, there is no reasoning, no planning, no path finding. The idea is to have an agent in a environment that changes fast, this is emulated but not parsing the

information from the server and using raw information about the map in the prompt.
- Deliveroo.js: explain the environment of the test
- GPT models: what they are, how they work
    - why GPT: i chose it because it's the most known, very powerful and the API have the necessary parameters
    - why not LLAMA or other open source/open weight models -> no logit bias, not logprobs, worse understanding of the prompt (no answering with the action)
    - OpenAI API: how I used it, with the necessary parameters (for example temperature was set at 0 for reproducibility and to get the "raw" performance of the LLM)

Chapter 3 - Agent development

Describe the evolution of my agent, from the first approach to the final one, highlighting the problems and why I changed every approach.

- First approach: very try-and-error. Started with parsing the server information and implementing 7 parameters one after one by checking in the test where the agent was having problems (like looping or choosing impossible actions) -> dropped because the development was going in the wrong direction, by introducing too much helps I was no more studying the LLM actual capabilities
- Second approach: stateless full raw. I just put the raw information from the server calls in the prompt, letting the LLM understand the map, the goal and everything by itself, but prompts were bad and the results were bad
- Final agent: new prompts following the literature, adding also the stateful approach (with the chat history), still bad results but more solid and reproducible and I started collecting data to investigate deeper
- Extra: "what is, between the neighboring cells, the closest one to the goal?" and "how do I reach it?" -> to see if the LLM is able to understand the map orientation: approach discarded due to some problems I want to discuss in the "result discussion" chapter

Chapter 4 - How data has been collected

Describe how the data has been collected, the tests done, the results collected, the heatmaps created, the data analyzed

- Attention visualizer in BERT: to show why the raw map may be a bit difficult to understand for the LLM due to all the characters like {, }, ecc. Also cite the tests done encoding the map (base64) to reduce the number of characters but it didn't reduced the number of tokens
- Prompts used: show the prompt and explain some choices, underlying also some decisions, like making the changing part of the prompt at the end of the prompt (the position of the agent that is updated every query) so that I could take advantage of the caching pricing of the API; also briefly discuss the effect of adding the result of an action in prompt (cheating because the LLM would answer based on its "math" skills and not on the actual map)
- Choices in prompt creation, accordingly to papers (give a role to the LLM, where to put the goal inside the prompt, KnowNo for the "give me only the action" part). The fact that stateful "works" may indicate that the structure of the prompt is correct
- Heatmap creation: how it's done, how to read them, how they are used, why it's useful

Results discussion

- Stateless: show cosa le heatmap hanno in comune se grandi o piccole (tipo l'avere problemi nelle celle orizzontali e verticali) cercando di dare una spiegazione
- Stateful: path visualization, show that with the increasing size of the map, the agent would have more problems in reaching the goal and due to the limitation of the number of tokens in the context (and the prompt being quite big due to it containing the map), wasting actions by looping would stop the system; evaluate some examples by comparing the path taken by the agent with the optimal path
- Stateless & Stateful: stateful works better due to "LLMs are few-shot learners" and the history of action-effect helps the LLM to understand the map and the goal; inoltre, come vediamo nella stateless, in certi casi l'agente si incanta in certi punti della mappa e lì le azioni rischiano di loopare
- Extra: closest cell: bias in the orientation of the map that can be seen from the heatmaps. The problem here is that, due to that bias, the LLM was prone to give the wrong tile as answer and then all actions would have moved the agent further away from the goal.
- Difference between different models (3.5, 4o, 4o-mini) -> 3.5 is the worst, 4o and 4o-mini are similar -> some benchmarks to show the best

Conclusions (2 pages)

> thesis objectives, achieved results, limitations, future developments

L'obiettivo era applicare un LLM da solo a questo problema. I risultati hanno dimostrato che dipende molto dalla dimensione della mappa e che, nonostante magari 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 finisce, ci rimane impantanato e poi i token finiscono.

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.

Limitations: prompt fatto seguendo la letteratura ma dio solo sa se è il prompt migliore. Il funzionamento del logit bias di open ai cambia 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 sicuramente un modello più potente in quanto si vede che effettivamente modelli migliori danno risultati migliori. Implementazione di RAG per aiutare nella decisione. Mettere mano a dei modelli open e testarli. Provare con modelli multimodali e mettere uno screenshot dell'environment nel prompt.