Prompt engineering

Ottimizzare i prompt per usare large language models.

Obiettivi:

- capire le possibilità e limitazioni di llms
- migliorare le performance del modello su vari tasks
- permette di interfacciare con llms e integrarlo con vari tools
- usare conoscenza esterna per permettere al llm di ragionare su queste risorse senza retrainare.

Writing good prompts

- Iniziare con simple prompts e fare prompt che man mano aggiungono cose per ottenere migliori risultati.
- Usare istruzioni specifiche all'INIZIO DEL PROMPT.
- Sii dettagliato e descrittivo per migliori output ma non troppo dettagliato
- Non includere informazioni eccessive, llm ha un attention windows limitata
- Usa esempi per guidare l'output, tune the behiavour.

Elementi di un prompt

- **Instruction** → cosa vogliamo dal modello
- **Context** → informazioni esterne o spiegazioni per garantire una migliore risposta, **è molto importante.**
- **Input data** → input o domanda al quale voglio risposta
- **Output indicator** → formato in cui voglio l'output – es: Sentiment: (mette l'output)

ESAME: POSSO DIRE DI NON RISPONDERE SE RILEVA CHE NON APPARTIENE AD NLP, OPPURE COME CONTEXT POSSO DARE LE SLIDE E LUI RISPONDERA' SOLO IN BASE ALLE SLIDE.

In-Context learning

Abilità di un LLM di performare un task interpretando il prompt sfruttando le informazioni di contesto date nel prompt non aggiornando i suoi parametri.

Nel prompt posso specificare: reference material, esempi di task per illustrare il pattern desiderato, posso specificare le istruzioni step by step, chiarire parti ambigue o template in cui mettere l'output.

Prompt enginnering si affida molto al **CONTEXT LEARNING** → **NON HO BISOGNO DI FINE TUNARE IL MODELLO PER TUTTI I TASK.**

Gli llms non sono ancora molto abili nel REASONING, si usano particolari tecniche per ottenere migliori risultati.

System Prompts

Do' un prompts una volta per guidare le successive interazioni.

Le successive risposte saranno influenzate da questo comportamento.

"You are a helpful and knowledgeable assistans.."

Tecniche di prompt engigneering

Zero-shot prompting → prompt senza includere esempi, sfrutta le sue informazioni

Few-shot prompting → dò alcuni esempi per guidare il modello in una risposta migliore. ES: Il modello capisce che sto cercando di usare una parola fittizia e risponde correttamente.

Limitazioni: è limitato quando c'è ragionamento complesso, non riesce né con fewshot né con zero-shot. Ho bisogno di ulteriori tecniche di prompt engigneering.

TECNICHE di prompting PER AIUTARE IL RAGIONAMENTO

Chain of thought prompting → permette ragionamenti complessi specificando gli step intermedi che dall'input permettono di arrivare all'output.

Posso combinarlo a few-shot prompitng per avere migliori risultati anche su task complessi.

**questa è un'abilita che nessuno si aspettava potesse avere un llm*

Self-consistency prompting → usa una chain of thought iterative, dò tante volte lo stesso prompt, considero le diverse risposte e prendo la risposta più frequente.

Meta prompting → guido il modello attraverso gli step logici per risolvere il problema, non concretamente ma in una maniera astratta (il contesto non è un esempio specifico ma un procedimento generale). Spiego il processo specificando per esempio le formule senza risultati. **IL CONTESTO E' IL PROCESSO SPIEGATO IN QUESTO CASO.**

TASK AGNOSTIC META-PROMPTING → E' stato dimostrato che aggiungendo la parola "ragiona step-by-step" forzo il modello a pensare ad un procedimento e quindi ottengo nuovi risultati. Identifica gli step e poi risponde. Devo dire **"inizia la risposta con lets think step by step". Poi posso definire anche meglio la struttura della risposta che voglio.**

Meta meta prompting → **chiedo al modello di generare un prompt per un problema particolare, poi uso questo prompt per rispondere al mio problema. Posso dire "genenera una procedura step by step" poi la uso nel prossimo prompt.**

**E' UN FIELD EMPIRICO, si scoprono le cose sperimentalmente*

Prompt chaining → Uso più prompt per ottenere il risultato che voglio, divido il task in parti specifiche con il prompt specifico. Prendo il risultato del primo e lo metto come risultato del secondo prompt fino a che non raggiungo il risultato finale.

Role Prompting → **chiedo al modello di impersonare un ruolo specifico mentre risponde, di solito miglioare l'accuracy perché si modifica di conseguenza il tono lo stile e la profondità di informazione. "You are a food critic".**

Structured prompting → divido il prompt in parti, e uso DELIMITATORI ###, ===, >>> per delimitare le parti. Posso uasre anche XML tags come delimitatori perché di solito gli llm sono allenati su web content <classes> </classes>.

Uno dei framework usato per fare structured promption è **COSTAR FRAMEWORK**, divide il prompting in diverse sezioni:

- context
- Objective
- Style
- Tone
- Audience, il pubblico a cui deve rivolgere la risposta
- response , formato della risposta

Generate knowledge prompting → prima usa l'llm per generare conoscenza relativa al task e poi la incorporo nel prompt per la risposta finale.

- 1° parte → genera knowledge particolare
- 2° parte → usa quello che hai generato per rispondere

Genera knowledge non inclusa nel training

Retrieval Augmented Generation – RAG

Combia llm con motori di ricerca, supera le limitazioni della llm accedendo a documenti aggiornati o dati specifici (per esempio aggiornati dopo il training, inoltre non c'è bisogno di modificare i pesi)

Uso la query per trovare queste informazioni nello search engine e poi li uso per rispondere.

Utile perché posso usarlo anche CON DOCUMENTI PRIVATI per costruire IL CONTESTO DINAMICAMENTE, A RUNTIME.

Si fa in 2 step.

- 1° step → ho i documenti, come libri
- 2° step → **divido i documenti in chunks, devo decidere quanto sono grandi, si genera l'embedding del chunk e vengono memorizzati in un vector DB.**

Quando faccio prompt prima cosa genero l'embedding, e metto il chunk come context e lui mi dà risposte simili.

molto richiesto perché risponde **basandosi su DATI PRIVATI, è il futuro dei CHATBOT*

ESAME

Esistono molte altre tecniche di prompt…

Prompt testing

Devo testare il prompt prima di usarlo nella mia applicazione → PROMPT TESTING TOOLS, fa vari tentativi per trovare la migliore struttura e il format del prompt (in base al modello)

Posso modificare vari parametri come la temperatura per controllare l'output style, tono o precisione

OpenAI playground, google ai studio, lm studio

LLM Settings

Accendendo ad un modello posso modificare vari parametri mentre progetto un prompt come la temperatura o **il top p** che aggiusta la diversità delle risposte limitando la scelta dei token con una determinata threshold. (Man mano si sceglie il next token secondo questa soglia, i token che possono essere la next word sommati devono fare massimo questa soglia, invece il top k prende i primi k e basta).

La temperatura invece aumenta la probabilità di base delle parole. Se t è bassa, 1/t è alto quindi la prob del primo valore diventa ancora maggiore.

**si usa perché a volte si paga per ogni token generato*

- **Stop sequences:** se viene generato questo token si interrompe la generazione
- **Frequency penalty:** riduce ripetizioni penalizzando il modello ogni volta che ripete una parola
- **Presence penalty**: penalizza il modello ANCHE SE LA PAROLA VIENE RIPETUTA 1 SOLA VOLTA.
- **Response format**: come mi aspetto l'output.

LM STUDIO

Ho accesso ad ogni hugging face model, posso usarlo anche come model server.