Un enfoque para mejorar el proceso de prompt engineering en aplicaciones basadas en LLM (Documento de trabajo)

V. Braberman[1], S. Uchitel[1], M. Carrasco[2], F. Mayr[2], and S. Yovine[2]

[1]DC, FCEyN, Universidad de Buenos Aires
[2]Universidad ORT Uruguay

Agosto 2024

Hoy en día muchas aplicaciones en la industria y en la academia que quieren utilizar grandes modelos de lenguaje como máquinas universales para responder preguntas recurren a la contextualización por medio de *prompts*. Los prompts son una pieza clave de este tipo de uso: tienen la intención de condicionar al modelo para que, en combinación con los mecanismos de *fine tuning* y *retrieval-augmented generation*, se generen respuestas que estén alineadas con el objetivo de la tarea en cuestión [1].

Identificar tareas y diseñar prompts efectivos es entonces una actividad fundamental para construir estas aplicaciones modernas. Hoy, la ingeniería de prompts está en su infancia: habitualmente es realizada de manera adhoc por parte de los programadores siguiendo un proceso como el esquematizado en la Figura 1.

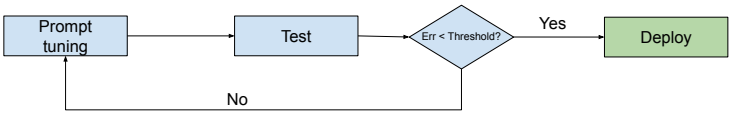

Figura 1: Proceso habitual de ingeniería de prompts.

También existen propuestas de métodos automatizados basados en algún tipo de feedback loop en el que un benchmark (casos de test) y una función de pérdida son usados en un esquema de optimización sobre el espacio discreto de los prompts, por ejemplo, Automatic Prompt Engineer (APE) [2] y Promptbreeder [3], o continuo de los embeddings como es el caso de Prefix-Tuning [4]. Si bien la evidencia empírica da cuenta de mejoras en la performance de la ta-

rea para los casos de test, estos enfoques tienen una serie de limitaciones para resolver varias cuestiones ingenieriles.

Lo primero a recordar es que un prompt, en realidad, afecta una distribución de probabilidad sobre el espacio de tokens posibles y, en definitiva, combinado con el mecanismo de decodificación da como resultado un programa probabilístico en dónde hay una distribución subyacente sobre los posibles resultados. Por lo tanto, basar un proceso de ajuste o análisis de prompts en los resultados obtenidos para cada caso de test oculta una realidad más compleja: un prompt define una distribución de probabilidad sobre el conjunto de los outputs posibles para cada caso de prueba y no un solo resultado [5].

Por otro lado, puede ser difícil tener un oráculo y/o una adecuada función de pérdida para una tarea particular. Además podría ser difícil definir el resultado esperado para casos fuera del conjunto de casos presentes en el benchmark y entender si y cuándo el modelo de lenguaje está generalizando de manera correcta.

Estas cuestiones nos deberían llevar a pensar en otras formas de evaluar y determinar si se está mejorando efectivamente la performance de una tarea. Un método como el del test *metamórfico* [6] debería ser aplicable, pero para ello hay que repensar cómo modelar el comportamiento de un modelo guiado por un prompt.

Para ello consideramos que el trabajo de aprendizaje de modelos subrogados, en particular autómatas probabilísticos, a partir de grandes modelos de lenguaje guiados [5], puede darnos una ventaja comparativa como base de ciertos tipos de análisis. Este tipo de objetos lograría capturar, para un prompt determinado y un conjunto (potencialmente infinito) de inputs de la tarea que se quiere realizar, caracterizado de manera formal, un modelo matemático que nos mostraría simbólicamente el espacio de "soluciones" generadas posibles para esos inputs.

El modelo formal así construido podría revelar características de la distribución que es susceptible de arrojar luz sobre el potencial comportamiento en tiempo de inferencia más allá de resultados generados por muestreo. Además, podría constituir un objeto accionable concreto para enunciar o verificar propiedades metamórficas, por ejemplo, el impacto en la distribución del lenguaje generado por el modelo de lenguaje dado por cambios semánticos controlados en el prompt. A partir de esta idea, pensamos que un proceso iterativo para la ingeniería de prompts como el ilustrado en la Figura 2, basado en herramientas desarrolladas en el marco del proyecto ANII IA_1_2022_1_173516 "Verificación de Sistemas Inteligentes con Componentes con Capacidad de Aprendizaje", podría conducir a resolver los problemas de los enfoques actuales identificados precedentemente.

Agradecimientos

Documento de trabajo del proyecto IA_1_2022_1_173516 "Verificación de Sistemas Inteligentes con Componentes con Capacidad de Aprendizaje" financiado por la Agencia Nacional de Investigación e Innovación (ANII), Uruguay.

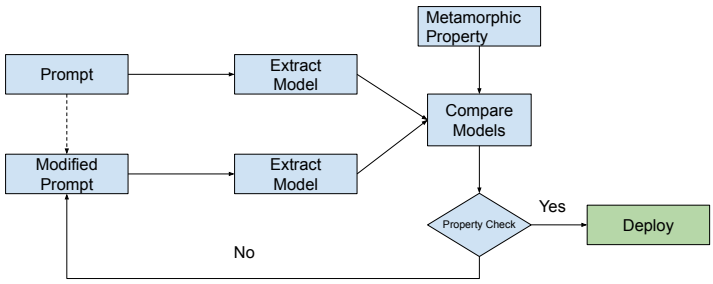

Figura 2: Enfoque propuesto.

Referencias

[1] L.-F. Bouchard y L. Peters, *Building LLMs for Production*. Towards AI, 2024.

[2] Y. Zhou, A. I. Muresanu, Z. Han et al., *Large Language Models Are Human-Level Prompt Engineers*, 2023. arXiv: 2211.01910 [cs.LG]. dirección: https://arxiv.org/abs/2211.01910.

[3] C. Fernando, D. Banarse, H. Michalewski, S. Osindero y T. Rocktäschel, *Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution*, 2023. arXiv: 2309.16797 [cs.CL]. dirección: https://arxiv.org/abs/2309.16797.

[4] X. L. Li y P. Liang, *Prefix-Tuning: Optimizing Continuous Prompts for Generation*, 2021. arXiv: 2101.00190 [cs.CL]. dirección: https://arxiv.org/abs/2101.00190.

[5] M. Carrasco, F. Mayr, S. Yovine et al., "Analyzing constrained LLM through PDFA-learning," *CoRR*, vol. abs/2406.08269, 2024. DOI: 10.48550/ARXIV.2406.08269. arXiv: 2406.08269. dirección: https://doi.org/10.48550/arXiv.2406.08269.

[6] S. Segura, D. Towey, Z. Q. Zhou y T. Y. Chen, "Metamorphic Testing: Testing the Untestable," *IEEE Software*, vol. 37, n.º 3, págs. 46-53, 2020. DOI: 10.1109/MS.2018.2875968.