Manual de Prompting para Large Language Models

¿Qué es un LLM?

Un Large Language Model (LLM) es un modelo de inteligencia artificial diseñado para comprender y generar lenguaje natural (Generación de texto, Traducción automática, Resumen automático, Respuesta a preguntas, Completar texto. Entre otros.), basado en enormes volúmenes de texto, aprendiendo patrones lingüísticos de un gran data set para predecir y generar respuestas coherentes y contextuales.

¿Qué es un prompt?

El prompting, en el contexto de los Large Language Models, se refiere a la práctica de proporcionar una entrada inicial o "prompt" al modelo para guiar su generación de texto. Un prompt puede ser una pregunta, una frase o incluso una palabra que actúa como disparador para que el modelo produzca una respuesta o continúe el texto de una manera coherente y contextualmente relevante. Los prompts eficaces son cruciales para obtener las respuestas deseadas y aprovechar al máximo las capacidades del modelo.

¿Cómo funcionan?

Los LLM predicen como continuar un texto, por lo tanto, hay un componente de aleatoriedad. Al generar texto, estos modelos pueden tomar decisiones diferentes, como seleccionar la siguiente palabra de una distribución de probabilidad en lugar de elegir siempre la opción más probable. Esto introduce variabilidad en las respuestas generadas para una misma entrada, resultando en diferentes posibles salidas cada vez que se utiliza el modelo.

En el siguiente ejemplo podemos ver un prompt, el cual usara el LLM para contestar al usuario. En las respuestas generadas podemos ver como a pesar de usar un mismo prompt y con el mismo mensaje de parte del usuario se pueden obtener resultados diferentes, aunque en ambos casos la primera predicción se la misma.

Ejemplo 1:

Prompt: “Completa las oraciones entregadas.” Usuario: Escribir es una tarea que requiere de ...
Respuesta 1: Escribir es una tarea que requiere de concentración, práctica y habilidades lingüísticas.
Respuesta 2: Escribir es una tarea que requiere de concentración, claridad de pensamiento y habilidad para organizar las ideas de manera coherente.

Buenas Prácticas de Prompting

En esta sección se agrupan las consideraciones y técnicas que se deben tener en cuenta siempre que se escriba un prompt.

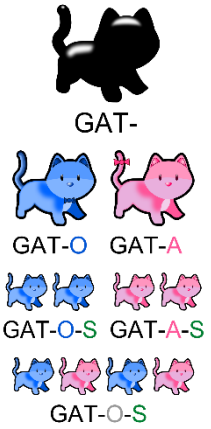
- Claridad y Precisión: Sea claro y específico en su solicitud para evitar ambigüedades. Use un lenguaje sencillo y directo.
- Brevedad y Concisión: Mantenga los prompts cortos para evitar sobrecargar al modelo. Elimine información que no contribuya a la tarea.
- Relevancia: Incluya solo información que sea relevante para la tarea en cuestión. Evite información que pueda desviar al modelo de la respuesta deseada.
- Proporcionar Contexto: Proporcione información de necesaria para que el modelo entienda el escenario. Ajustando el nivel de detalle del contexto a la complejidad de la tarea.

Considere también que los LLM están entrenados con grandes cantidades de datos, así que tienden a aprender el lenguaje generalizado y sus estructuras más estandarizadas, por lo que también le es más fácil entender los prompts que están escritos de esta manera. Por un lado, la ortografía puede hacer una gran diferencia, evitando que el LLM se confunda con nuestras instrucciones y mejorando la consistencia en la escritura, pero también debemos tener otros aspectos de la lingüística:

- Semántica:
  "Es el *estudio del sentido de las palabras, el estudio de la función de las palabras, función que consiste en transmitir un sentido*. (Guiraud: *La Semántica*)"[1]. En nuestro contexto se relaciona fuertemente con la claridad, precisión y proporcionar un contexto, eligiendo adecuadamente el vocabulario a usar.
  - Frase con mala semántica: "Voy a plantar un libro en el jardín para cosechar conocimiento."
  - frase con semántica mejorada: "Voy a leer un libro en el jardín para cultivar mi conocimiento."

  Esta corrección mantiene la metáfora del crecimiento y la naturaleza, pero utiliza "leer" que es la acción apropiada para obtener conocimiento de un libro, y "cultivar" que puede referirse metafóricamente al desarrollo personal o educativo.

- Morfología:
  " Es la parte de la gramática que se ocupa de la estructura de las palabras, las variantes que estas representan y el papel gramatical que desempeña cada segmento en relación con los demás elementos que la componen. En el estudio de la morfología se analizan las raíces, afijos, prefijos, sufijos, inflexiones y la formación de palabras. La morfología ayuda a entender cómo se construyen las palabras para expresar



[1] https://www.ucm.es/plataformaele/la-semantica

diferentes significados, tiempos, cantidades, y otros aspectos gramaticales."[2] Como en la imagen.

- Sintaxis:
  "Parte de la gramática que estudia el modo en que se combinan las palabras y l os grupos que estas forman para expresar significados, así como las relaciones que se establecen entre todas esas unidades."[3]

  - Frase con mala sintaxis: "El gato negro rápido saltó sobre la mesa dormido estaba."
  - Corrección de la frase: "El gato negro y rápido saltó sobre la mesa donde estaba dormido."

  Aunque la selección de las palabras le da al modelo el contexto de la tarea, es una correcta sintaxis la que le "proporciona" la relación entre las ideas.

- Pragmática:
  "La pragmática se define como una parte de la lingüística que centra su estudio en la **relación existente entre la lengua, los hablantes y las situaciones** en las que se produce la comunicación. Abarca todos aquellos campos no lingüísticos como tal, pero que sí interfieren en la comunicación."[4]

  Por lo general los prompts no son eficientes fuera de los contextos en los que fueron diseñados, incluso cumpliendo todas las características de un buen prompt. Por ejemplo, el uso de prompts en inglés es usualmente recomendado, ya que representa la mayoría de los datos con lo que fueron entrenados estos modelos, sin embargo, para tareas en español esto puede llegar a generar inconsistencias en la escritura, incluso siendo la traducción una de las habilidades de los LLM más usados. Incluso el uso de modismos direcciona la generación al uso de vocabulario determinado, por lo que aplicarlos sobre contenidos que no tienen las mismas características perjudicara los resultados.

  Por lo mismo, debemos tener en cuenta las condiciones con las que los LLM fueron entrenados. En el caso de GPT-3.5 Turbo, uno de los LLM más usados a la fecha, fue entrenado con datos previos a septiembre de 2021, lo que impide que pueda responder por información, nombres, expresiones, etc. creadas o difundidas por internet posterior a esa fecha[5]. Para más información mire (Agregar sección comparativa de LLMs).

---

[2] https://www.ucm.es/plataformaele/morfologia#:~:text=La%20morfolog%C3%ADa%20es%20la%20parte,dem%C3%A1s%20elementos%20que%20la%20componen

[3] https://dle.rae.es/sintaxis

[4] https://www.unir.net/educacion/revista/pragmatica-lenguaje/#:~:text=La%20pragm%C3%A1tica%20se%20define%20como,s%C3%AD%20interfieren%20en%20la%20comunicaci%C3%B3n.

[5] El leguaje y tendencias de los usuarios cambian constantemente, por lo que los LLM necesitan ser actualizados periódicamente.

Dependiendo del caso, quizás deba tener en cuenta otros aspectos (o áreas) de la lingüística, si quiere ver más información sobre lingüística computacional. En cualquier caso, recuerde que el objetico del análisis lingüístico de un prompt es para conseguir una escritura lo más "estandarizada" que sea posible, lo que facilitará la tarea al LLM.

- Ventanas de Contexto:
  Como podrás comprobar por tu cuenta, los LLM tienen dificultades para realizar tareas sobre contenidos extensos.
  La longitud máxima del texto que puede recibir como entrada para realizar una tarea, se conoce como tamaño de la ventana de contexto de un LLM[6]. El contexto incluye el prompt, ejemplos y cualquier contenido que adicionemos al mensaje inicial. Modelos como GPT-4 tienen ventanas de contexto de 4096 tokens que corresponde a 3000 palabras aproximadamente, lo que implica que, si le pedimos al modelo que procese una ventana de contexto mayor, el LLM solo considerara las ultimas 3000 palabras.

  Pero incluso cumpliendo estos parámetros, en la actualidad, los modelos generativos en general (Texto, Imágenes, Audio, Video), cargan con el problema de "La aguja en el pajar"[7]. En particular en los LLM el problema se puede ver así: Piensa en un libro relativamente extenso, como "el Quijote", al cual en una pagina al azar se ha escondido la siguiente oración, "La comida favorita de Luis es la Mojarra" (un nombre y un plato que no aparecen en la obra original), Si un lector humano leyese nuestra versión del libro, seguramente seria capaz de responder a la pregunta "¿Cuál es la comida favorita de Luis?". Sin embargo, un LLM se "perdería" en las generalidades del Libro, con otros nombres, platos etc., que aparecen más frecuentemente, o con más relevancia, en el texto. Este comportamiento se acentúa con la longitud del texto por lo que es posible que, si le entregamos solamente la página modificada, pueda resolverlo.

  Insisto, la ventana de contexto incluye toda la información entregada, por lo que un prompt muy largo, muchos ejemplos etc., también puede generar este comportamiento.

Uso de Ejemplos

Para tareas complejas que requieren de salidas especificas o muy estructuradas, puede ser necesario un contexto más extenso, en estos casos es útil incluir ejemplos de la tarea deseada. Asegúrese de que el ejemplo esté directamente relacionado con la tarea solicitada, es claro y representativo.

- Zero-Shot Learning: Se refiere a la capacidad del modelo de realizar una tarea sin haber visto ejemplos previos.
- One-Shot Learning: Implica proporcionar un único ejemplo para guiar al modelo en la realización de una tarea.

---

[6] https://medium.com/@amiraryani/what-is-your-context-window-772b1c65b881

[7] Este articulo explica el problema más a fondo y muestra los resultados de un modelo generativo que promete resolver/reducir este problema gemini_v1_5_report.pdf (storage.googleapis.com).

- Few-Shot Learning: Incluir un número muy limitado (3-5, max 10) de ejemplos para que pueda generalizar y realizar la tarea deseada con eficacia.

Prompt:

Ejemplo Usuario:

Ejemplo Respuesta:

Respuesta Real (sin):

Respuesta Real (con):

Recomendaciones:

1. Los ejemplos en Few-Shot deben ser variados, una forma de lograrlo es representar los distintos casos en los que el prompt sin ejemplos suele tener problemas.
2. A pesar de ser una técnica relativamente fácil de implementar y con resultados muy buenos, al usarla debemos tener presente que aumentar el contenido/palabras que se le entregan al modelo, también crecen los recursos necesarios y por tanto el costo de su uso.

Ejemplo 3: [8]

Programación Incremental (Incremental Programming)

Realice la tarea en pasos, proporcionando o solicitando información adicional según sea necesario. Use la retroalimentación del modelo para ajustar el prompt y dirigir la tarea.

- Uso de Roles o Personajes: Asigne roles específicos al modelo para guiar su comportamiento (asistente, experto, crítico, profesor …). Asegúrese de que el rol esté alineado con la tarea solicitada. También puede ser útil describir personajes detallados y pedirle al modelo que los personifique.
- Ofrecer recompensas:
- Razonamiento:
- Delimitadores: Uso de “señales” como guiones, asteriscos o signos de mayor/menor. Los delimitadores ayudan a distinguir las diferentes partes de un proceso. Por ejemplo, para señalar el inicio y el fin de un input o output.
- Formatos (Input-output):
- Negative prompting:

[8] https://www.promptingguide.ai/techniques/fewshot

Like Programming (Avanzado):

Embeddings (Avanzado):[9]

Flujos con prompts (Avanzado)

Permita que el modelo construya conocimiento a través de interacciones sucesivas. Un flujo con prompts implica una secuencia de interacciones en la que cada prompt (indicación o pregunta) toma en cuenta la respuesta previa, permitiendo así una conversación o proceso continuo que construye sobre el conocimiento previamente compartido o generado.

Tomando la notación usada en 2305.10601.pdf (arxiv.org), $\mathrm{p}_\theta(\mathrm{x})$ representa el resultado producido por un LLM "p", con hiperparámetros "θ" y la secuencia de texto "x". Las estrategias anteriormente nombradas, en las que se usa solamente un prompt, se conoce como Input-output prompting (IO) y se escribe como $\mathrm{p}_\theta^{prompt}(\mathrm{x})$ ó $\mathrm{p}_\theta^{IO}(\mathrm{x})$.

Para problemas en los que el "IO" es insuficiente se recomienda dividir la tarea en subtareas. Los flujos de prompts son paso a paso en el que se usan varios prompts y/o procesamientos organizados de tal manera que produzcan el resultado deseado.

CoT (Chain of Thought): es un método que implica descomponer el proceso de pensamiento en pasos lógicos **secuenciales**. En la interacción con modelos de lenguaje, se utiliza para resolver problemas complejos o preguntas detalladas, mostrando explícitamente el razonamiento detrás de la respuesta[10].

Formalmente podemos definir $\mathrm{p}_\theta^{CoT}(\mathrm{x}) = \mathrm{p}_{\theta\mathrm{n}}^{IO}(\mathrm{x}, \mathrm{z}_1, \dots, \mathrm{z}_\mathrm{n})$, donde $\mathrm{z}_1 = \mathrm{p}_{\theta 1}^{IO}(\mathrm{x})$ y $\mathrm{z}_\mathrm{i} = \mathrm{p}_{\theta\mathrm{i}}^{IO}(\mathrm{x}, \mathrm{z}_1, \dots, \mathrm{z}_{\mathrm{i}-1})$ para 1< i <n. Es decir, usar prompts mas enfocados a partes específicas de la tarea, encadenándolos hasta llegar al resultado, teniendo en cuenta completa o parcialmente los resultados anteriormente generados.

CoT - Self Consistency: Ya que los LLM pueden generar respuestas distintas de una misma cadena de entrada, a pesar de poder realizar una tarea exitosamente en un intento, puede que no lo haga en todos. Para este tipo de casos los prompts detallados y el uso de los hiperparametros es importante, pero no siempre suficientes.

La técnica de CoT-SC nos permite reducir esa variabilidad en los resultados a costa de realizar mas consultas al LLM para comparar sus resultados y elegir el resultado mas frecuente o adecuado.

---

[9] https://platform.openai.com/tokenizer

[10] Si bien la propuesta original considera la descomposición de la tarea dentro de un mismo prompt, note que esto sería similar al incremental programing, pero el procesamiento por separado en distintos prompts permite al modelo concentrarse en cada parte de la tarea lo que la hace una técnica mucho más poderosa.

Su versión más simple usa “n” llamados al LLM con IO prompting y se toma la respuesta más repetida, $\mathrm{p}_{\theta}^{COT-SC}(\mathrm{x}) = \mathrm{argmax}_{\mathrm{y}}\#\{\mathrm{i} \mid \mathrm{p}_{\theta \mathrm{i}}^{IO}(x) = \mathrm{y}\}$, pero $\mathrm{p}_{\theta \mathrm{i}}^{IO}(x)$ puede ser remplazado por $\mathrm{p}_{\theta \mathrm{i}}^{CoT}(x)$, como se ve en la imagen, y la selección por “votos” puede ser remplazado por otro tipo de selección o agrupamiento de los resultados.

Valga aclarar que aunque la técnica pueda ser utilizada para reducir la variabilidad de los resultados de un prompt o un CoT, es recomendable que los prompts y/o bifurcaciones se construyan con aproximaciones distintas al problema, esto permite aprovechar mejor las capacidades del modelo al permitirle razonamientos distintos sobre el mismo problema.

ToT (Tree of Tougths):

Para tareas especialmente complejas o que requieren soluciones específicas un CoT-SC puede seguir siendo inadecuando, ya que puede ser que en la mayoría de sus ramas no consiga el resultado deseado, heredando los errores al resultado final, además de ser ineficiente en el uso de recursos.

La técnica de ToT pretende solucionar estos aspectos del CoT-SC realizando una búsqueda en un árbol de prompts aprovechando idas de algoritmos de búsqueda como DFS o BFS. Esto resulta en un proceso con mucha mas complejidad pero que reduce los recursos y permite encontrar repuestas más precisas. ----------

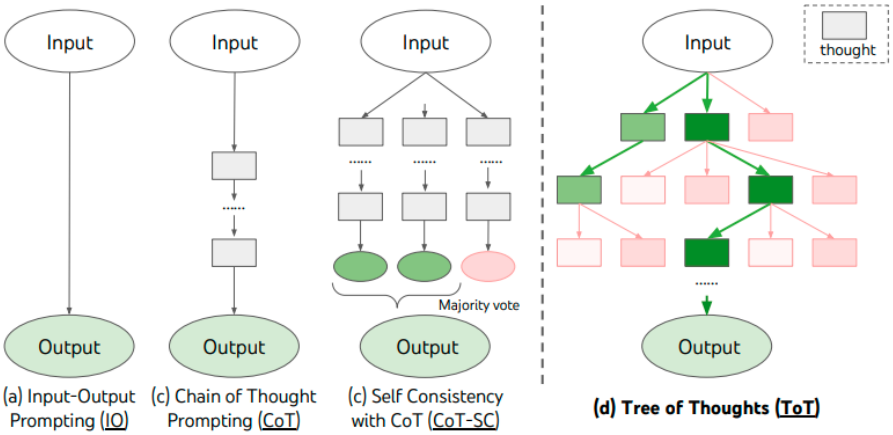


Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2,4,6.

Aprendizaje reforzado y Reflexion Agents (Avanzado)

[2303.11366] Reflexion: Language Agents with Verbal Reinforcement Learning (arxiv.org)

https://arxiv.org/pdf/2303.00001.pdf

RAG (Retrieval Augmented Generation):

El enfoque RAG combina un sistema de recuperación de información con un modelo generador de texto. RAG es altamente adaptable y permite ajustar su conocimiento interno de manera eficiente, sin necesidad de volver a entrenar el modelo por completo.

RAG funciona al tomar una entrada y recuperar un conjunto de documentos relevantes de una fuente dada (por ejemplo, bases de datos académicas). Estos documentos se utilizan como contexto junto con el mensaje de entrada original y se envían al generador de texto para producir el resultado final. Esta característica hace que RAG sea especialmente útil en situaciones donde la información puede cambiar con el tiempo. Esto es crucial ya que el conocimiento estático de los modelos de lenguaje puede resultar insuficiente. RAG permite a los modelos de lenguaje evitar la necesidad de reentrenamiento, lo que les permite acceder a la información más actualizada para generar resultados confiables mediante la generación basada en la recuperación.

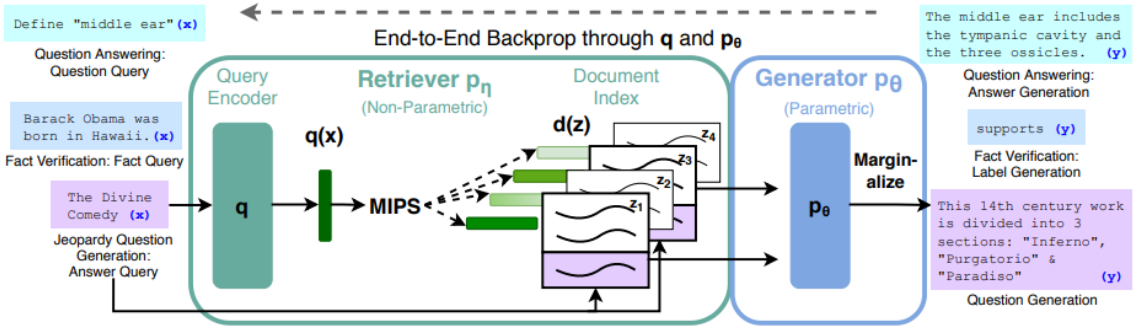


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Fuente: Lewis et al. (2021)

Directional Stimulus Prompting

El estímulo direccional es una técnica bastante avanzada que ayuda a los modelos de lenguaje grande (LLM) a dar respuestas más precisas y relevantes. ¿Cómo funciona? Bueno, básicamente consiste en darle al modelo indicaciones específicas que lo dirijan hacia la información más importante de un texto.

Imaginemos que cada vez que el modelo recibe una pregunta o una tarea, le damos una especie de guía o dirección sobre qué es lo más importante que debe tener en cuenta. Esta guía se adapta a lo que necesitamos en ese momento. Por ejemplo, si queremos que el modelo resuma un texto largo, le damos indicaciones sobre las palabras clave o los temas principales que debería incluir en el resumen.

Para que esta técnica funcione bien, hay algunos pasos que seguimos:

- Definimos la indicación: Esto es como darle al modelo una hoja de ruta, le decimos claramente lo que queremos que haga, ya sea enfocarse en un tema en particular o seguir una estructura específica.
- Generamos el contenido: Utilizamos esa indicación para dirigir al modelo hacia los aspectos más importantes del texto. Así nos aseguramos de que el resultado sea relevante y preciso.
- Limitamos el alcance: Establecemos un límite de palabras para mantener el resumen corto y preciso.

Por ejemplo, podríamos decirle al modelo que resuma un artículo centrándose solo en los personajes principales y las acciones que realizan. O podríamos pedirle que destaque los conceptos clave del texto. Hay muchas formas de guiar al modelo, dependiendo de lo que necesitemos.

Esta técnica no solo hace que los modelos de lenguaje sean más precisos, sino que también los hace más útiles en entornos académicos y profesionales. Al usar el estímulo direccional, podemos estar seguros de obtener resultados que sean tanto precisos como relevantes para lo que estamos buscando.

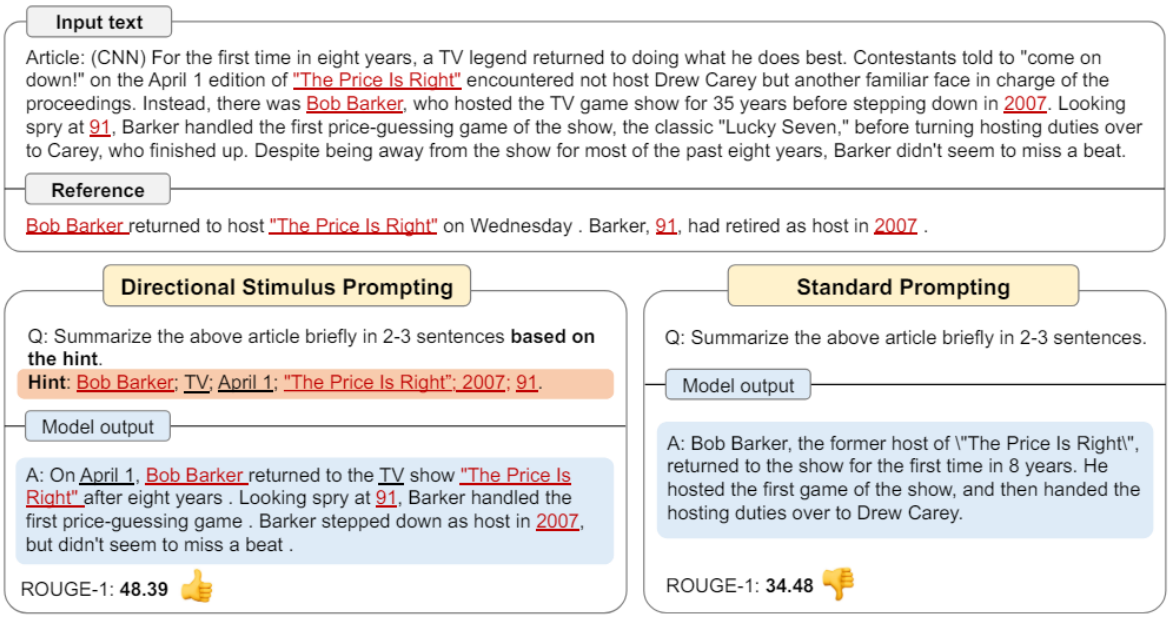


Figura 3: comparación de la técnica de estímulo direccional y la técnica estándar que utiliza LLM como ChatGPT para la tarea de resumen. Fuente: Zekun Li et al. (2023)

Bibliografía

- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

- Li, Z., Peng, B., He, P., Galley, M., Gao, J., & Yan, X. (2024). Guiding large language models via directional stimulus prompting. Advances in Neural Information Processing Systems, 36.