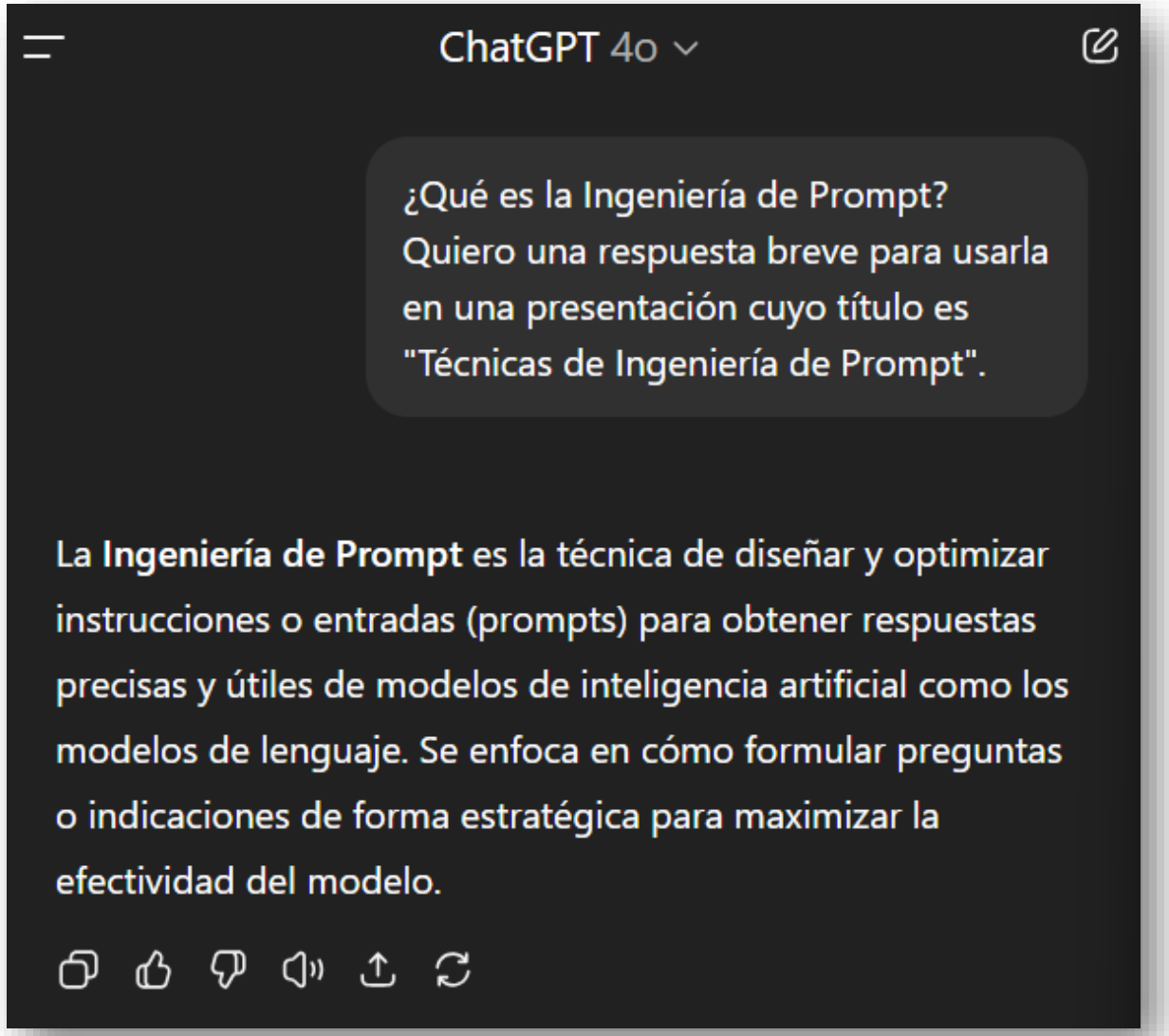
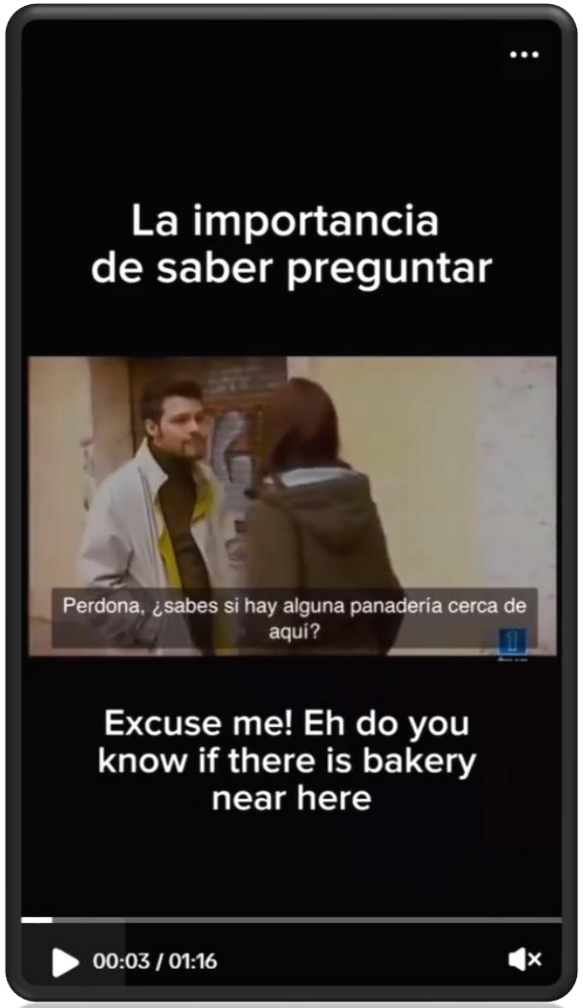
Técnicas de Ingeniería de *Prompt*

Juan Bernardo Quintero

¿Qué es la Ingeniería de *Prompt*?

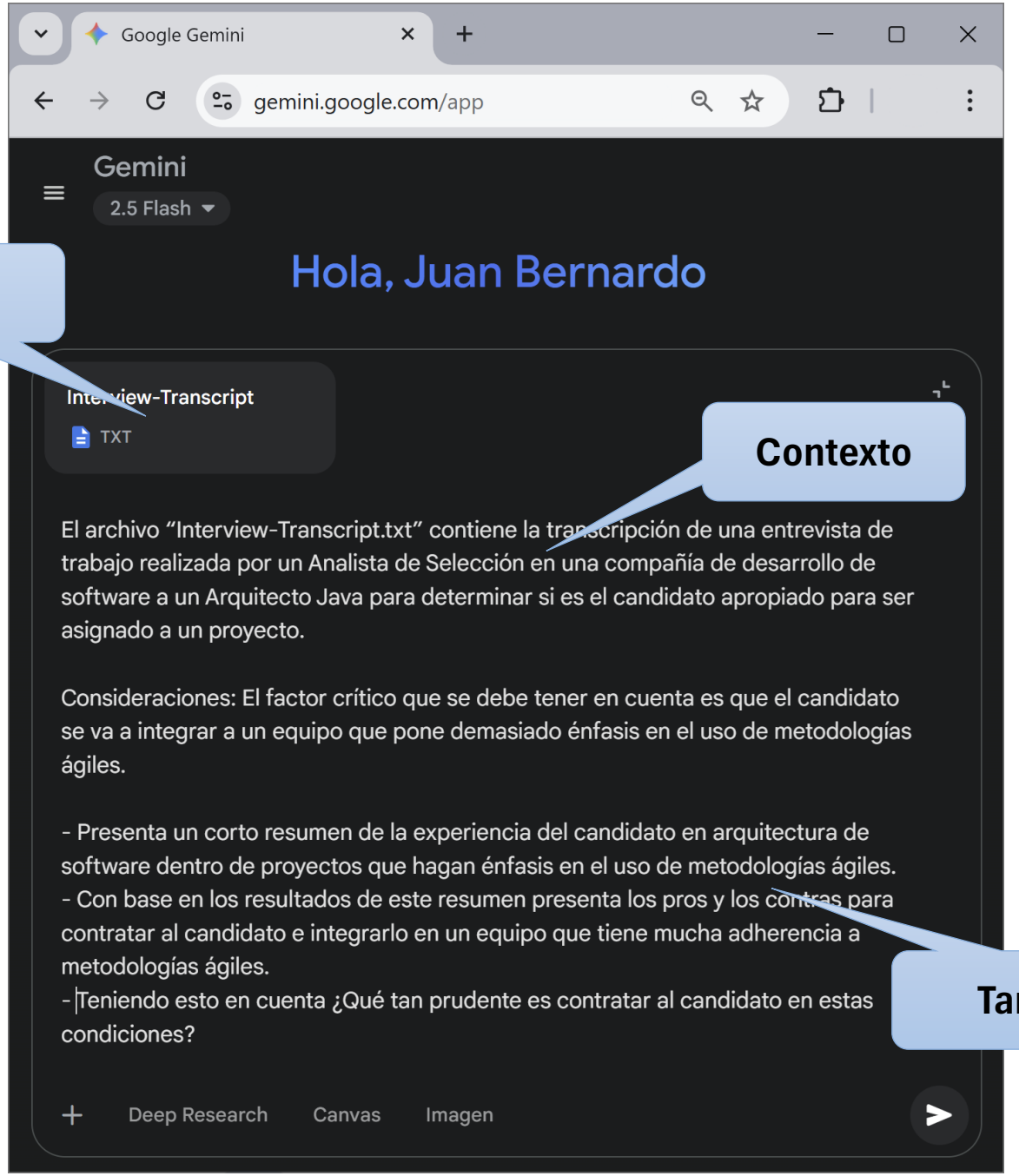




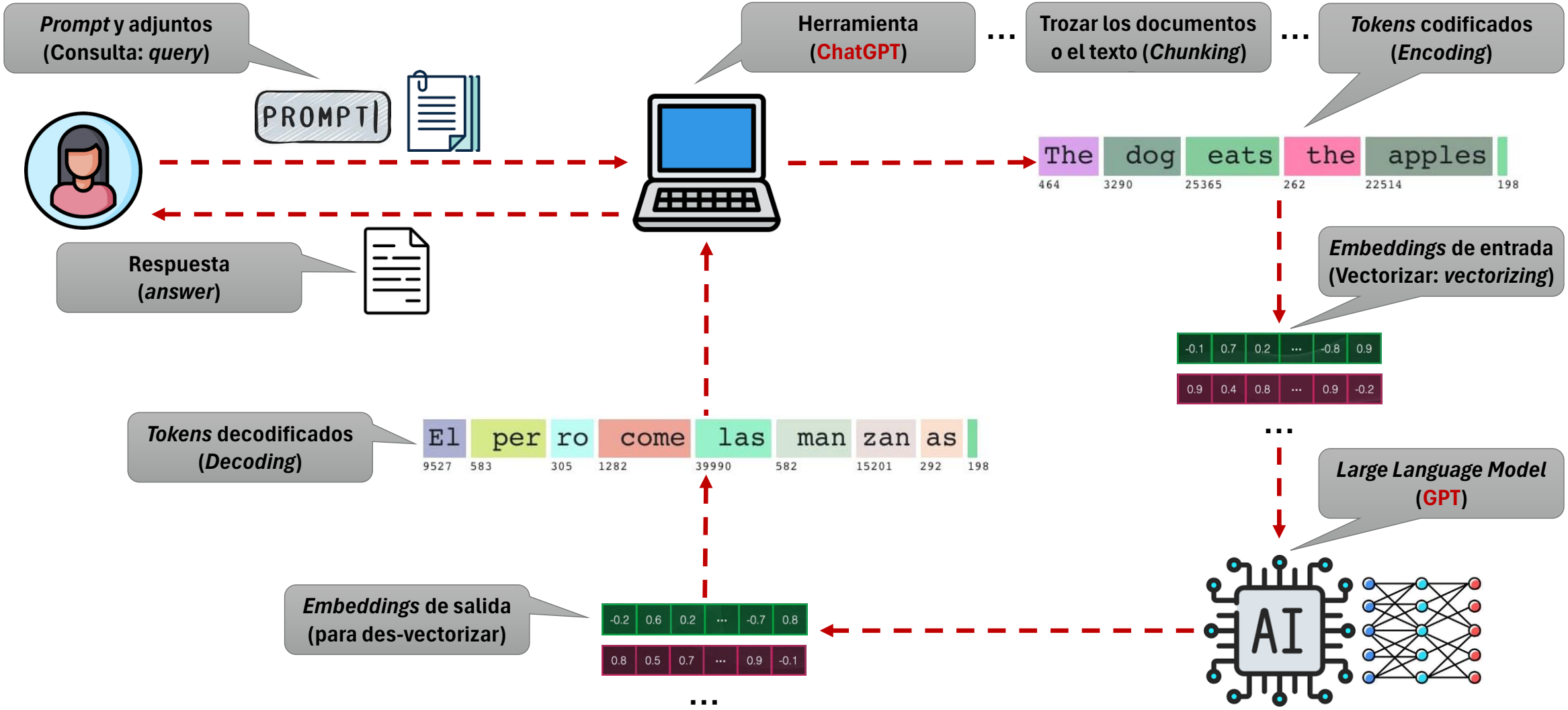
PROMPT ENGINEERING FRAMEWORKS

¿Qué estructura debe tener un *Prompt*?

¿Qué es lo básico que debe tener un *Prompt*?

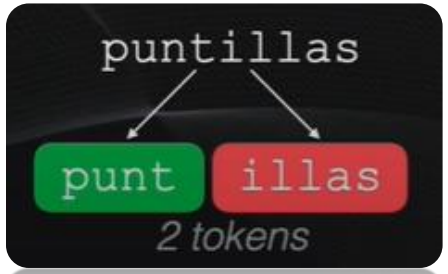


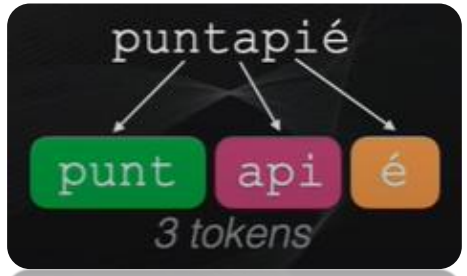
Herramientas de IA Gen: Funcionamiento



¿Qué son los *Tokens*?

- Un token es una **unidad de texto** que se utiliza durante el procesamiento del modelo.
- En la mayoría de los casos, un token se corresponde con una palabra individual, aunque en algunos casos también puede representar una **sub-palabra** o un **carácter**.
- El número de caracteres promedio por token depende del idioma:
  - Inglés: **7** caracteres/token
  - Español: **9** caracteres/token
  - Chino: **12** caracteres/token





Adaptado de: https://www.youtube.com/@codificandobits

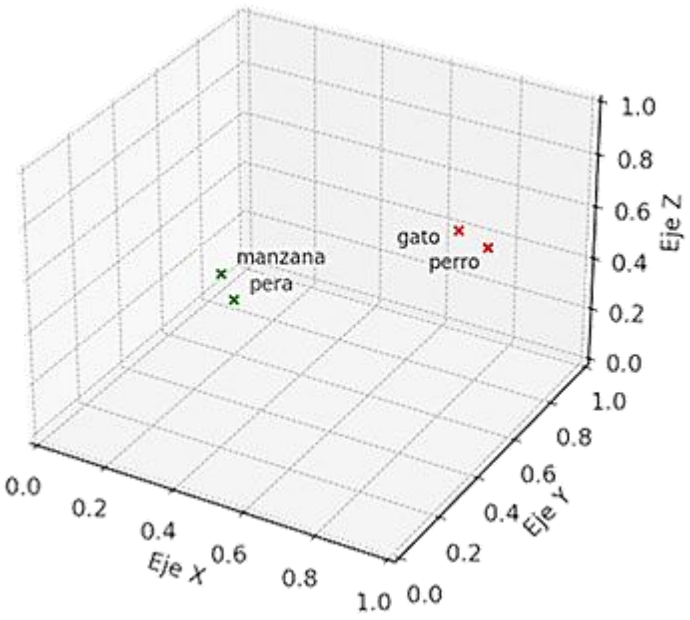
¿Qué es la "*Tokenización*"?

- La "**tokenización**" es el proceso de convertir el texto en tokens, considera espacios en blanco, números e incluso emojis.
- Implica el uso de un vocabulario: a cada token se le asocia un **número equivalente** para ser procesado por la red neuronal *Transformer*.
- Una vez procesados, los tokens son convertidos **de vuelta a texto** para poder ser interpretados por **humanos**.

Adaptado de: https://www.youtube.com/@codificandobits

¿Qué son los *Embeddings*?

- Las redes neuronales **no procesan realmente tokens**, estos son tan solo una representación inicial de las palabras.
- Después de **convertir** las palabras en **tokens**, se deben pasar a ***embeddings***.
- Los *embeddings* son una **representación vectorial** de las palabras, que son lo que realmente procesa la **red neuronal *Transformer***.



Adaptado de: https://www.youtube.com/@codificandobits

“Attention is all you need”

- Las redes neuronales *Transformer* permiten generar *embeddings* que son capaces de capturar la información del contexto.
- A esta capacidad se le conoce como “**Mecanismo Atencional**” y permite que una palabra tenga un significado particular dentro de un contexto específico.
- En el ejemplo, a pesar de que la palabra “banco” es la misma, sus *embeddings* son considerablemente diferentes debido a su contexto.

Adaptado de: https://www.youtube.com/@codificandobits

¿Qué es “Ventana de Contexto”?

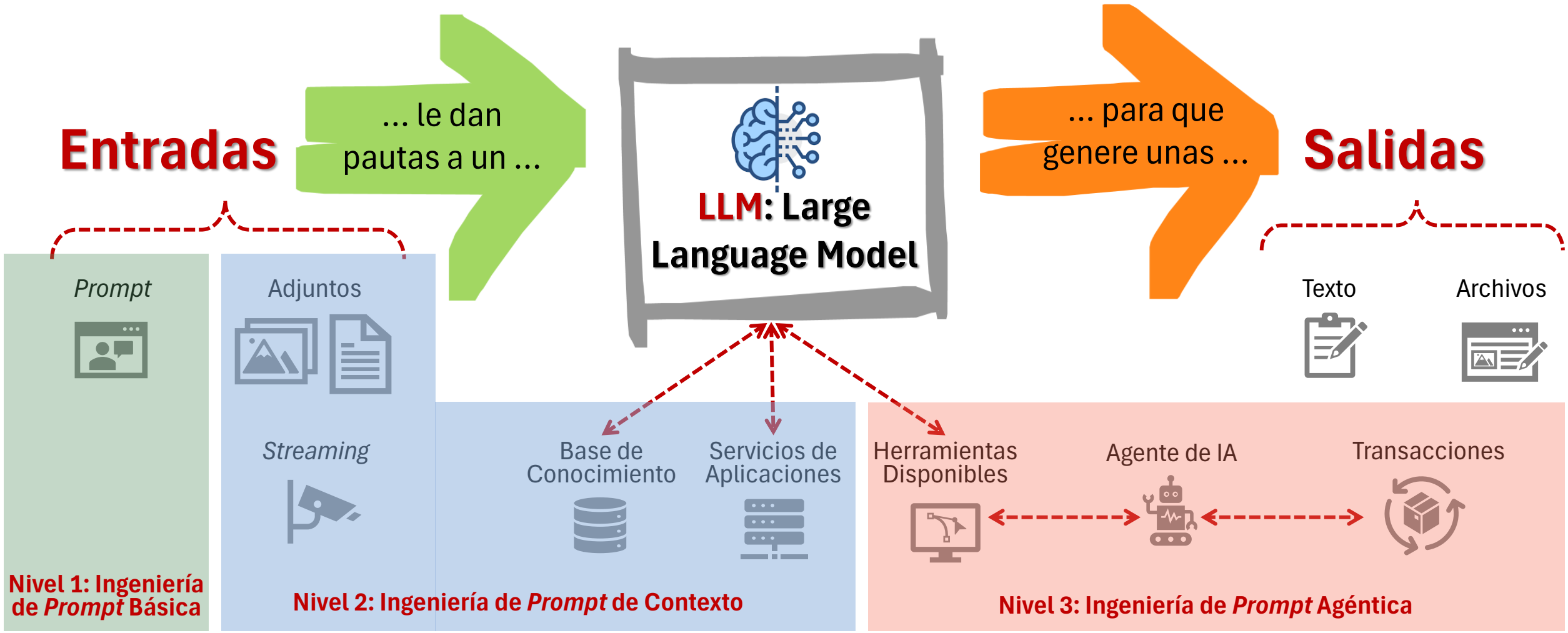
- La ventana de contexto se refiere a la cantidad de tokens que un gran modelo de lenguaje puede procesar al momento de interpretar una secuencia de texto. Para el cálculo de la ventana de contexto se consideran tanto los tokens de **entrada**, los de **salida** y los de **razonamiento**.

- Para **GTP 4o** la ventana de contexto es de **128.000 tokens**, alrededor de **96.000 palabras** en español, entre **275 y 385 páginas** dependiendo del formato

- Para **Gemini 2.5 Flash** o **Claude 4** es de **1’000.000 tokens**, alrededor de **750.000 palabras** en español.

Adaptado de: https://www.youtube.com/@codificandobits

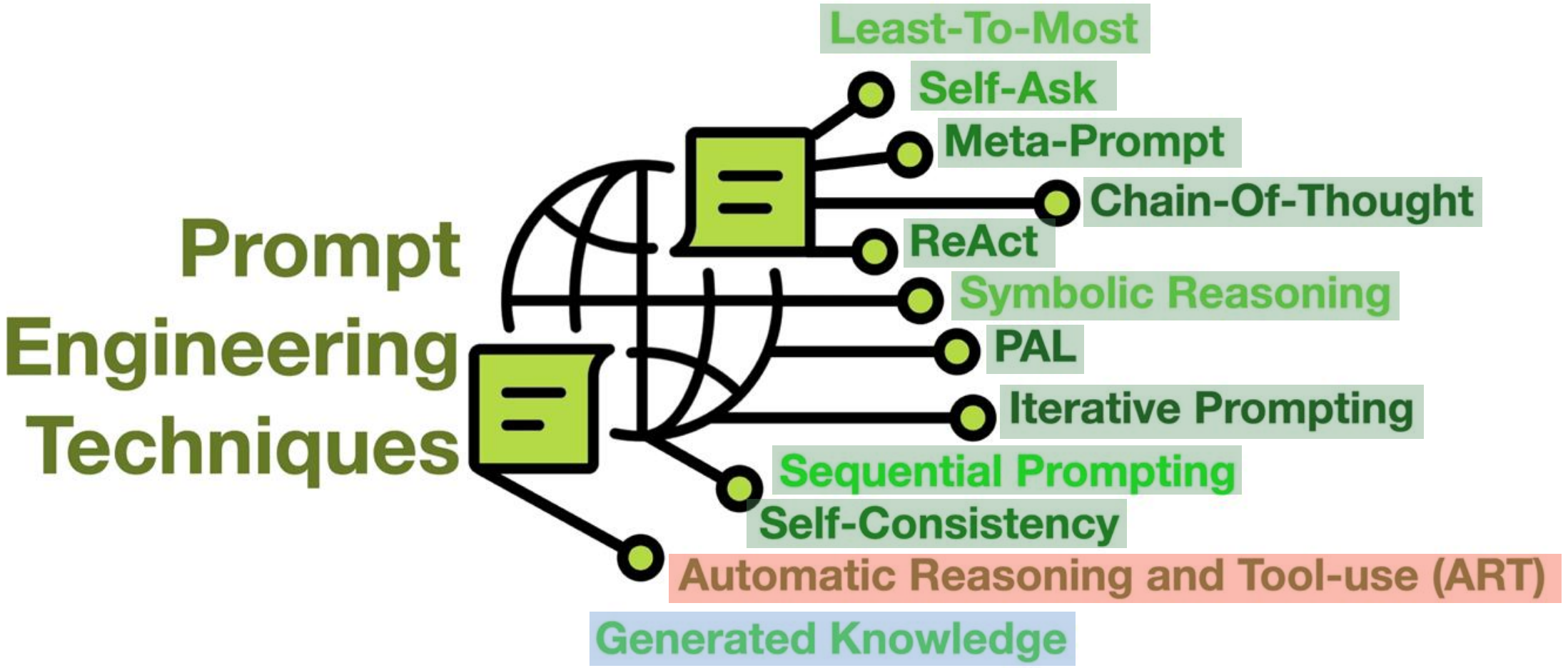
Evolución de la Ingeniería de *Prompt*



Niveles de *Prompting*

Nivel	Enfoque principal	Técnicas / Artefactos	Perfil que lo usa
1. Básico	Claridad del *prompt*: se concentra en la forma del mensaje que entra al modelo grande de lenguaje.	• *Few-shot prompting* • *Chain-of-thought (CoT)* • Formato de salida	Usuario avanzado o creador de contenido que conversa directamente con un LLM.
2. Contexto	Selección del corpus contextual: diseñar qué información acompaña al *prompt* dentro de la ventana de contexto.	• Aplicaciones *RAG (Retrieval-Augmented Generation)* • Diseño de asistentes	Desarrollador de productos basados en LLM que necesita respuestas fiables en producción.
3. Agéntico	Autonomía y orquestación: los LLM actúan como agentes autónomos que planean, llaman herramientas y se corrigen a sí mismos.	• Diseño de agentes • MCP (*Model Context Protocol*) • *Tool-calling* • *Guard-rails*	Arquitecto de soluciones IA, creadores de agentes corporativos o de flujos que automatizan tareas.

Técnicas de Ingeniería de *Prompt*



https://www.humanfirst.ai/blog/12-prompt-engineering-techniques

Caso de Estudio: Asistente de IA para RRHH

Técnicas para escribir los *prompts* de un Asistente de IA para RRHH que analiza la transcripción de una entrevista de trabajo.

Enunciados y *prompts* disponibles en https://github.com/juanbdo/prompt-engineering-lab

Prompting de menos a más

- **Contexto**: El archivo "interview-transcript.txt" contiene la transcripción de una entrevista de trabajo realizada por un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: El factor crítico que se debe tener en cuenta es que el candidato se va a integrar a un equipo que pone demasiado énfasis en el uso de metodologías ágiles.

Least-To-Most Prompting	**Descompone un problema complejo en subproblemas más simples y los resuelve secuencialmente.**

Posible *Prompt*:

1. Inicialmente presenta un corto resumen de la experiencia del candidato en arquitectura de software dentro de proyectos que hagan énfasis en el uso de metodologías ágiles.
2. Con base en los resultados de este resumen presenta los pros y los contras para contratar al candidato e integrarlo en un equipo que tiene mucha adherencia a metodologías ágiles.
3. Teniendo esto en cuenta ¿Qué tan prudente es contratar al candidato en estas condiciones?

Prompting de auto-pregunta

- **Contexto**: Se va a realizar una entrevista de trabajo por parte de un **Analista de Selección** en una compañía de desarrollo software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: Cómo dedo orientar la entrevista si el candidato se va a integrar a un equipo que pone demasiado énfasis en el uso de metodologías ágiles.

Self-Ask Prompting	**Permite al LLM razonar explícitamente y descomponer la pregunta en sub-preguntas de seguimiento.**

- **Posible *Prompt***: ¿Qué preguntas podrían hacerse para determinar si el candidato es apto para integrarse a un equipo que tiene mucha adherencia a metodologías ágiles?
- ¿La idea que tiene el candidato del uso de metodologías ágiles es positiva o negativa? justifica la respuesta / ¿Qué tanta experiencia tiene el candidato en arquitectura de software y metodologías ágiles? / ¿Cuáles son los pros y los contras para contratar al candidato e integrarlo en un equipo que tiene mucha adherencia a metodologías ágiles? / ¿Consideras que es prudente contratar al candidato en estas condiciones?

Meta-*prompting*

- **Contexto**: Se va a realizar una entrevista de trabajo por parte de un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un cliente "*scrum lover*".
- **Consideraciones**: Se tienen dudas de qué criterios de evaluación usar y qué preguntas realizarle al candidato para que el cuestionario esté alineado con los criterios de evaluación.

Meta-Prompting	**Utiliza un *prompt* general para guiar al LLM a reflexionar sobre su propio rendimiento y modificar sus instrucciones en consecuencia.**

- **Posible *Prompt***: Podrías redactar un *prompt* que sirva para encontrar los cinco criterios de evaluación más apropiados para un cliente "*scrum lover*" (mucha adherencia a esta metodología ágil) y luego redacta un cuestionario de diez preguntas que esté alineado con los criterios de evaluación que formulaste.
- Define una escala Likert que permita calificar el nivel de cumplimento del criterio por parte del candidato, de tal forma que pueda ser usado posteriormente para otros candidatos.

Prompting de cadena de pensamiento

- **Contexto**: El archivo "interview-transcript.txt“ contiene la transcripción de una entrevista de trabajo realizada por un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: El archivo es bastante voluminoso y contiene mucha información en la que se muestra el orden en el que el entrevistador y el entrevistado se comunicaron.

Chain-of-Thought Prompting	**Divide una tarea o problema más grande en subtareas, y luego encadenamos estas subtareas, usando el resultado de una subtarea como entrada para la siguiente subtarea. Desarrolla capacidades de razonamiento sofisticado.**

- **Posible *Prompt***: Podrías resumir la entrevista enfatizando los tópicos del tema de arquitectura Java que se abordaron y cuál es la experiencia y el conocimiento que tiene el candidato de cada uno; presenta únicamente el resumen y ten en cuenta que luego el resumen será utilizado para continuar con el proceso de análisis del candidato. Genera la respuesta como un archivo *.txt descargable.

ReAct (Razonar & Actuar)

- **Contexto**: El archivo "interview-transcript.txt" contiene la transcripción de una entrevista de trabajo realizada por un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: Se contratará al candidato para un cliente de mucha adherencia a Scrum, con base en su entrevista ¿Cuáles pasos se deben tener en cuenta para su proceso de *on-boarding*?

***ReAct* (*Reasoning & Acting*)**	**Combina el razonamiento con la toma de acciones, permitiendo al LLM actualizar planes de acción en función de información adicional recopilada.**

- **Posible *Prompt***: Con base en la entrevista, qué frentes de la metodología Scrum debe fortalecer el candidato para ser contratado para un cliente de mucha adherencia a Scrum, ¿Cuáles pasos se deben tener en cuenta para su proceso de *on-boarding*? Define un plan de *on-boarding* para ser aplicado con el candidato de un mes de duración.

Razonamiento simbólico y PAL

- **Contexto**: El archivo "interview-transcript.txt" contiene la transcripción de una entrevista de trabajo realizada por un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: Se debe calificar porcentualmente la entrevista de este y otros candidatos considerando 3 dimensiones: scrum (50%), experiencia (30%) y herramientas (20%).

Symbolic Reasoning and PAL	**Permite convertir el razonamiento en un programa (*Program-Aided LLM*) que implica análisis de símbolos, colores, tipos de objetos, etc.**

- **Posible *Prompt***: Con base en la entrevista, genera una tabla de calificación de la entrevista considerando tres dimensiones: scrum (50 puntos), experiencia (30 puntos) y herramientas (20 puntos), poniendo para cada dimensión la justificación de cada calificación.

Prompting iterative

- **Contexto**: El archivo "interview-transcript.txt" contiene la transcripción de una entrevista de trabajo realizada por un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: Se necesita una lista de la experiencia del candidato detallada, mostrando para cada caso el área de experiencia y el nivel (Alto / Medio Alto / Medio / Medio Bajo / Bajo).

Iterative Prompting	**Asegura que las indicaciones sean contextuales, contengan ejemplos de entrenamiento (*few-shot*) y el historial de la conversación.**

- **Posible *Prompt***: Basándose exclusivamente en el archivo de trascripción de la entrevista, genera una tabla de experiencia detallada mostrando el nivel del candidato a un área específica; por ejemplo:
  - Pruebas : Alto.
  - Microservicios : Medio Alto.
  - Bases de Datos: Medio

Prompting secuencial

- **Contexto**: El archivo "interview-transcript.txt" contiene la transcripción de una entrevista de trabajo realizada por un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: Calcular la velocidad a la que habló el candidato en palabras por minuto.

Sequential Prompting	**Suele usar un pipeline para generar candidatos (elementos, componentes o técnicas relevantes) y luego hacer un ranking (para determinar los mejores elementos o su orden), considerando que usar LLMs a gran escala es costoso.**

- **Posible *Prompt***: Basándose exclusivamente en el archivo de trascripción de la entrevista, calcula la velocidad a la que habló el candidato en palabras por minuto con los dos siguientes métodos:

  1. Totalizado: Multiplicar el tiempo total de la entrevista por el total de palabras que habló el candidato y dividir por el total de palabras que se hablaron en la entrevista (tanto el candidato como el entrevistador).

  2. Por dialogo: Para cada dialogo del candidato contar las palabras y dividir por el tiempo extraído de la marca de tiempo del respectivo diálogo; finalmente promediar para todos los diálogos del candidato.

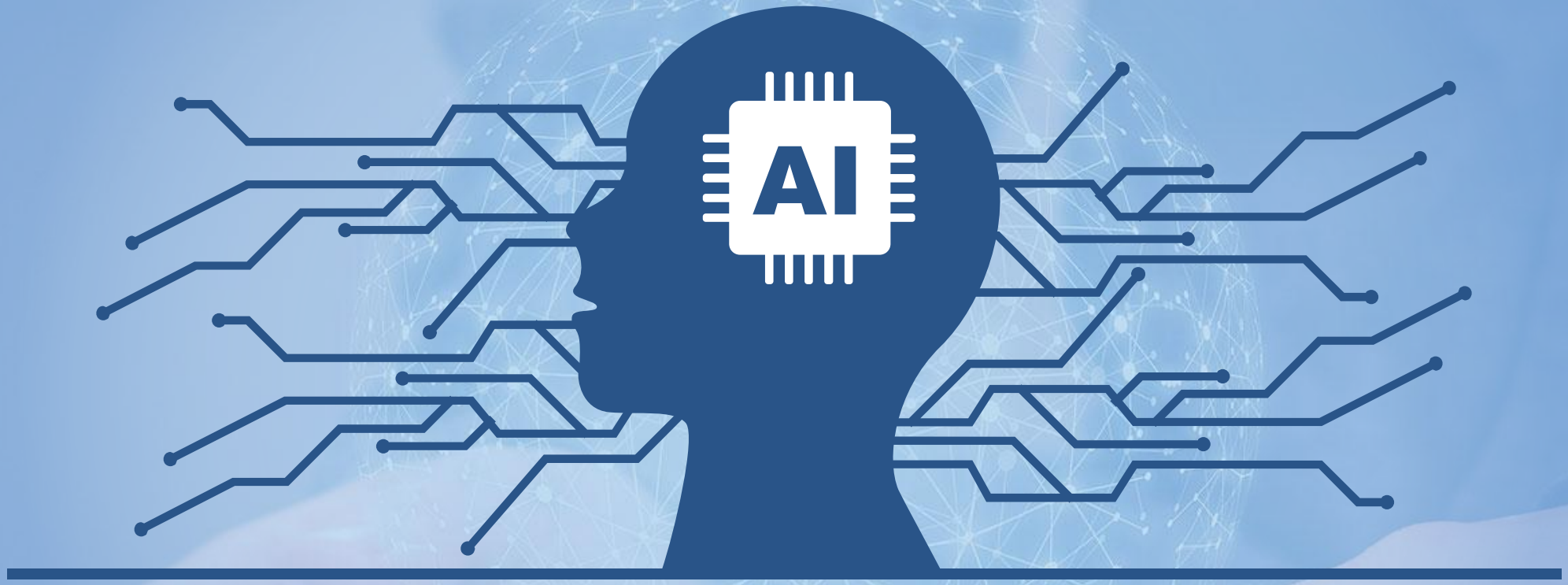
  ¿Cuál de los dos métodos es mejor?

Autoconsistencia

- **Contexto**: El archivo "interview-transcript.txt" contiene la transcripción de una entrevista de trabajo realizada por un **Analista de Selección** en una compañía de desarrollo de software a un **Arquitecto Java** para determinar si es el candidato apropiado para ser asignado a un proyecto.
- **Consideraciones**: Dejar que el LLM defina cual es el mejor método para calcular la velocidad a la que habló el candidato en palabras por minuto.

Self-Consistency	**Genera múltiples caminos de razonamiento y selecciona el más consistente como respuesta final.**

- **Posible *Prompt***: Basándose exclusivamente en el archivo de trascripción de la entrevista, propón los métodos que se pueden usar para calcular la velocidad a la que habló el candidato en palabras por minuto.
- Realiza el cálculo con cada uno de los métodos que propusiste y muestra los resultados
- ¿Cuál de los métodos es mejor?



¡ Gracias !