Evaluación jerárquica de modelos de lenguaje en diagnóstico clínico: superando la saturación mediante un pipeline multicapas (simplificado)

Análisis metodológico de PV0-PV4

Autores: Yago Mendoza, Javier Logroño
Institución: Foundation 29
Fecha: 20 de julio de 2025
Versión del Documento: 1.0

Este documento presenta un resumen de los hallazgos y la evolución de los marcos de evaluación para la herramienta de diagnóstico asistido por IA DxGPT, destinado a un público no especializado en el ámbito técnico.

Glosario de términos

Término	Descripción simplificada
LLM (Large Language Model)	Un modelo de inteligencia artificial avanzado, entrenado con enormes cantidades de texto para comprender, resumir, generar y predecir contenido. Es la tecnología base de herramientas como ChatGPT.
Pipeline de evaluación	Una secuencia de pasos o un flujo de trabajo automatizado diseñado para medir el rendimiento de los modelos de IA de forma sistemática y reproducible.
Juez-IA (LLM as a Judge)	El uso de un LLM muy avanzado para evaluar y puntuar las respuestas de otros modelos de IA, basándose en su comprensión del contexto y la lógica.
Sistema de supervisión semántica (BERT)	Un componente de IA especializado en determinar la similitud de significado entre dos textos, incluso si no usan las mismas palabras. Evalúa si dos diagnósticos son conceptualmente equivalentes.
ICD-10 / SNOMED CT	Sistemas de codificación médica estándar utilizados a nivel internacional para clasificar enfermedades y procedimientos. Proporcionan una base objetiva para comparar diagnósticos.
Posición en la lista (P1, P5, etc.)	La posición en la que aparece el diagnóstico correcto dentro de la lista de 5 propuestas generadas por el modelo. Una posición baja (ej. P1) indica una mayor çonfianza clínica"del modelo.
PV0, PV2, PV3, PV4	Versiones sucesivas de nuestro pipeline de evaluación. Cada versión fue un intento de mejorar la anterior, culminando en PV4, el sistema final y más robusto.

Resumen

Este informe describe la evolución de nuestros métodos para evaluar cuatro modelos de lenguaje de OpenAI sobre **450 casos pediátricos**. El análisis inicial reveló que los sistemas de evaluación simples generaban resultados engañosos. Un método basado en códigos médicos penalizaba diagnósticos muy precisos, mientras que un evaluador basado en una IA (un "Juez-IA") tendía a valorar todas las respuestas como buenas si eran plausibles, ocultando las diferencias reales de rendimiento entre los modelos.

Para solucionar esto, desarrollamos **PV4**, un sistema de evaluación avanzado que combina la **objetividad de los códigos médicos** con el **juicio contextual de una IA**. Además, introdujimos una métrica clave: **la posición del diagnóstico correcto en la lista de propuestas**, como un indicador de la confianza del modelo.

Esta nueva metodología reveló una jerarquía de rendimiento clara. El modelo **o3** emergió como el **más fiable y estable**, con la mejor posición promedio y el mayor número de aciertos en primer lugar. Aunque el modelo **o3-pro** obtuvo una tasa de aciertos brutos ligeramente superior, lo hizo a costa de una peor priorización de sus diagnósticos. En resumen, la calidad de la evaluación es tan importante como la calidad del modelo evaluado para obtener una visión precisa de sus capacidades.

1. Planteamiento del problema de evaluación

La validación de sistemas de IA para el soporte al diagnóstico clínico es un desafío crucial. No se trata simplemente de verificar si una IA .ªcierta", sino de evaluar la calidad, robustez y relevancia clínica de su razonamiento. Una hipótesis diagnóstica que suena plausible pero no es precisa puede comprometer la seguridad del paciente y la confianza del profesional.

Nuestro objetivo es establecer un marco de evaluación que vaya más allá de métricas superficiales y permita discernir con claridad el rendimiento real entre diferentes modelos de IA. Esta evaluación debe capturar no solo el acierto, sino también la priorización y la precisión de las propuestas diagnósticas. Este informe documenta el proceso que hemos seguido para construir dicho marco, aprendiendo tanto sobre los modelos como sobre nuestras propias herramientas de medición a partir de un dataset diverso (cuya composición se detalla en el **Anexo A**).

2. Evolución hacia un sistema de evaluación robusto

Para encontrar la mejor manera de medir el rendimiento de los modelos, desarrollamos una serie de "pipelines.º flujos de trabajo de evaluación. Cada uno corregía los defectos del anterior, llevándonos a un sistema final mucho más fiable.

La Tabla 2 resume este viaje metodológico. Esta progresión es la clave para entender por qué fue necesario desarrollar un sistema tan sofisticado como PV4. Es crucial entender que PV0, PV2 y PV3 fueron etapas de aprendizaje que nos permitieron identificar sesgos y limitaciones. **El único sistema válido y utilizado para los resultados finales es PV4**.

Cuadro 2. Evolución de los pipelines de evaluación: de la simplicidad a la robustez.

Pipeline	Ventajas	Inconvenientes	Lección aprendida
PV0	Simple y rápido.	Ciego al contexto clínico. Confundía similitud de palabras con relevancia diagnóstica.	La similitud textual por sí sola es una métrica engañosa.
PV2	Introduce objetividad usando códigos médicos estándar.	Demasiado rígido. Penalizaba respuestas correctas pero más específicas que el diagnóstico de referencia.	La codificación estricta no captura la flexibilidad del lenguaje clínico.
PV3	Un "Juez-IA"que entiende el contexto y los matices clínicos.	Demasiado "generoso". Hacía que todos los modelos parecieran igual de buenos, ocultando diferencias.	Un juicio puramente contextual premia la plausibilidad, no la precisión.
PV4	**Equilibra la objetividad de los códigos con la flexibilidad del juicio de IA**. Usa la posición como métrica de confianza.	Mayor complejidad interna.	La evaluación de alta fidelidad exige combinar objetividad, contexto y prioridad.

Los pipelines intermedios sufrían de "saturación de la tarea": un fenómeno donde modelos de distinta calidad parecían rendir igual. Los sistemas rígidos (PV2) castigaban la precisión, y los sistemas flexibles (PV3) eran demasiado generosos, premiando la plausibilidad en lugar de la exactitud. Esto nos obligó a diseñar un sistema híbrido que superara ambas limitaciones.

3. Lógica operativa de PV4: un sistema jerárquico

PV4 es el resultado de este aprendizaje. Es un sistema jerárquico que combina la objetividad de los códigos con la inteligencia del análisis contextual. Su principio fundamental es evaluar las 5 propuestas diagnósticas de un modelo en orden (de la 1 a la 5) y **detenerse en cuanto encuentra el primer acierto válido**. La puntuación del caso se basa en la posición de ese primer acierto, premiando así la confianza clínica del modelo.

El flujo de decisión para cada diagnóstico propuesto, como se muestra en la Figura 1, sigue una cascada de lo más objetivo a lo más interpretativo:

Paso 1: Verificación por códigos médicos. Primero, el sistema busca una coincidencia objetiva usando las clasificaciones estándar **SNOMED** e **ICD-10**. Si se encuentra una coincidencia directa o jerárquicamente cercana (ej. un sub-diagnóstico), la propuesta se valida y el proceso termina.

Paso 2: Supervisión semántica. Si no hay coincidencia de código, un **sistema de supervisión semántica** comprueba si el diagnóstico propuesto es conceptualmente equivalente al correcto, aunque se escriba con palabras diferentes.

Paso 3: Juicio de una IA experta. Si los pasos anteriores fallan o generan dudas, se recurre a un **Juez-IA** (un modelo de lenguaje avanzado) para que realice una evaluación

contextual final y determine si la propuesta es clínicamente válida.

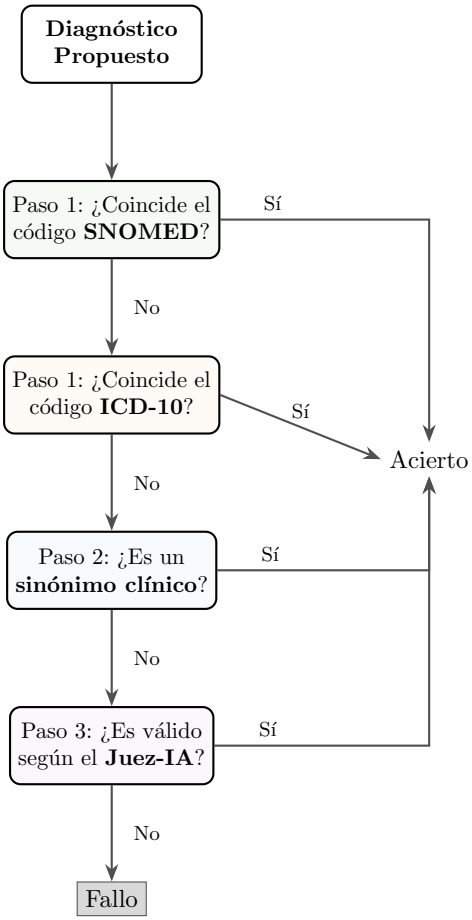

*Figura 1. Diagrama de flujo simplificado del proceso de evaluación de **PV4** para un único diagnóstico propuesto.*

4. Resultados y ranking final de modelos

La aplicación del pipeline PV4 nos permitió discriminar el rendimiento de los modelos con alta precisión. La Tabla 3 muestra los resultados finales. Aunque **o3-pro** tiene la tasa de acierto global más alta, **o3** demuestra ser el modelo más **confiable**: obtiene la mejor posición promedio y acierta en primer lugar con mayor frecuencia.

*Cuadro 3. Ranking y métricas clave de rendimiento por modelo (pipeline **PV4**).*

Métrica	o3	o1	o3-pro	4o
Tasa de acierto (%)	93.7 %	91.4 %	**96.4 %**	94.3 %
Posición promedio	**1,47**	1,59	1,60	1,63
Aciertos en posición 1 (P1)	**311**	305	299	299
Aciertos en posición 5 (P5)	**9**	17	24	20

La Figura 2 visualiza esta diferencia clave. Muestra cómo **o3** concentra la mayoría de sus aciertos en las primeras posiciones, mientras que otros modelos, aunque aciertan, lo hacen con menor prioridad. Esta capacidad de priorizar correctamente es fundamental para la utilidad clínica. Análisis más detallados sobre la estabilidad de los modelos y el desglose de métodos se encuentran en el **Anexo B**.

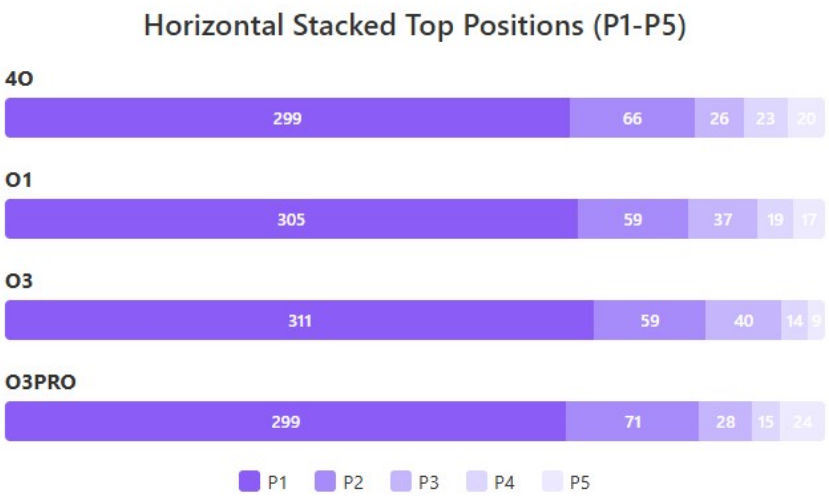

*Figura 2. Distribución de los aciertos en las Top 5 posiciones por modelo. **o3** destaca por su alta concentración de aciertos en primera posición (P1).*

5. Conclusiones

El diseño de un sistema de evaluación robusto es tan importante como el desarrollo de los propios modelos de IA. Nuestro proceso iterativo nos ha llevado a las siguientes conclusiones:

Veredicto del rendimiento de los modelos

- **o3** destaca por su **fiabilidad**: obtiene la mejor posición promedio (1,47) y el mayor número de aciertos en primera posición, lo que indica una mayor confianza clínica.
- **o3-pro** logra la **máxima cobertura** (96,4 % de aciertos) pero a costa de una peor priorización en su lista de diagnósticos diferenciales.
- La estabilidad de **o3** fue notablemente superior a la de otros modelos como **4o**, haciéndolo más predecible y consistente para un uso práctico.

Lecciones sobre la metodología de evaluación

- Los sistemas de evaluación simples pueden ser engañosos. La rigidez de los códigos (PV2) castiga la especificidad, mientras que la flexibilidad de un Juez-IA (PV3) oculta las diferencias de rendimiento.
- La mejor aproximación es un **sistema híbrido y jerárquico (PV4)** que combina la objetividad de los códigos con el juicio contextual, utilizando la **posición en la lista** como un indicador clave de la confianza del modelo.
- El evaluador no es un observador pasivo; su diseño define activamente qué se considera un "buenrendimiento.

A. Composición del dataset de evaluación

El dataset final de 450 casos se construyó a partir de un universo de 9.677 casos clínicos. El siguiente diagrama de Sankey (Figura 3) visualiza el flujo de Extracción, Transformación y Carga (ETL) que garantiza la diversidad y representatividad del conjunto de pruebas.

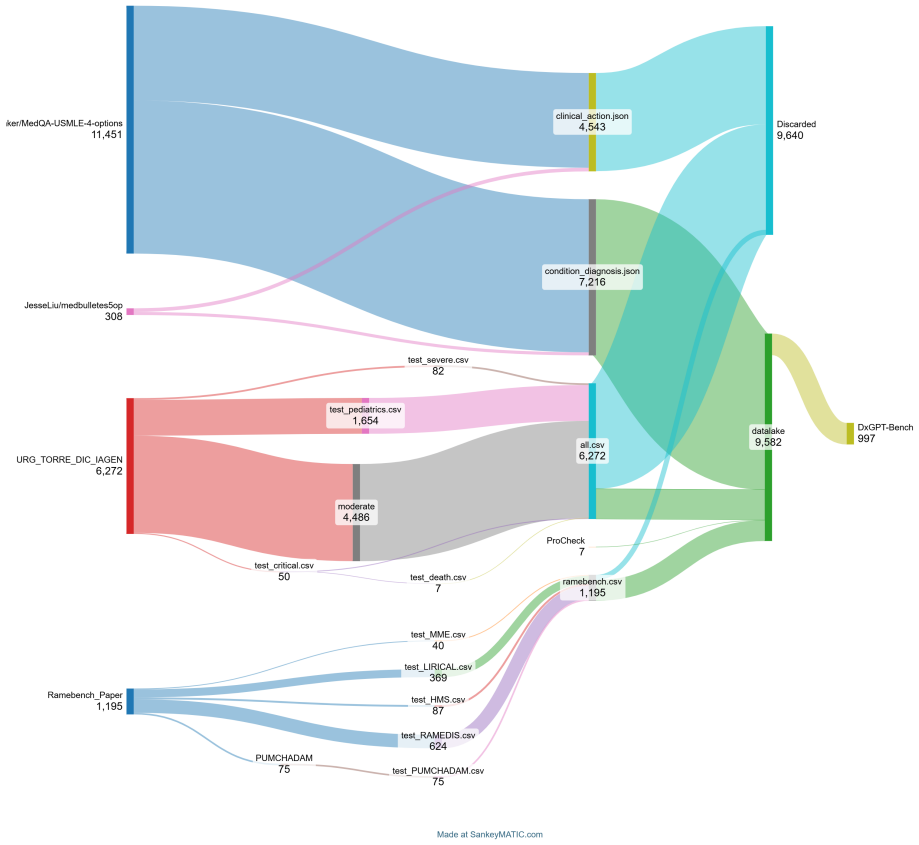

Figura 3. Diagrama de Sankey que visualiza el proceso de ETL para la composición del dataset de evaluación.

B. Figuras y tablas adicionales

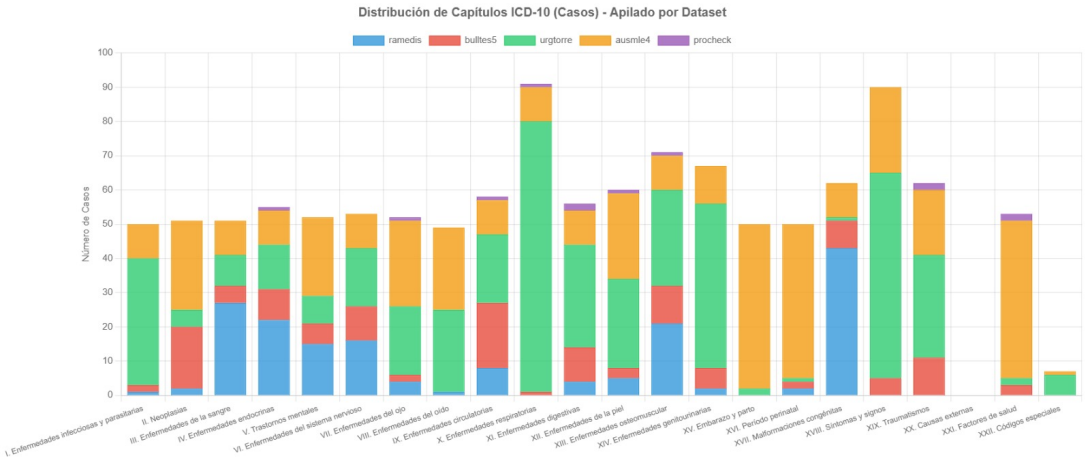

*Figura 4. Distribución de los 450 casos clínicos por capítulos de la codificación **ICD-10**.*

*Figura 5. Número de casos resueltos y posición promedio por cada método del pipeline **PV4**. Los tonos más oscuros representan mejor puntuación (posición más baja).*

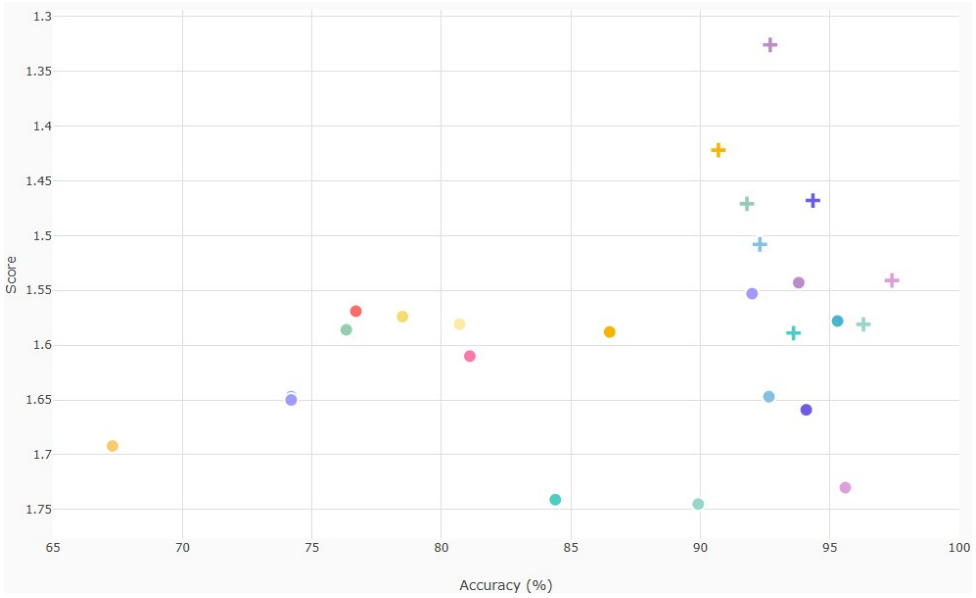

*Figura 6. Comparativa de rendimiento entre diferentes variantes de prompts (las cruces indican ejecuciones con **o3** y los círculos con **4o**).*

*Cuadro 4. Análisis de estabilidad del rendimiento entre **o3** y **4o**.*

Modelo	**Nº prompts (n)**	**PPos media ($\mu \pm \sigma$)**	**% acierto ($\mu \pm \sigma$)**
o3	7	1.474 ± 0.083	$93.65\,\% \pm 2.46$
4o	17	1.629 ± 0.066	$84.31\,\% \pm 8.91$

C. Prompts de alto rendimiento

A continuación se presentan los prompts que obtuvieron los mejores resultados para los modelos **o3** y **4o**, junto con sus puntuaciones de rendimiento (posición promedio y tasa de acierto).

C.1 Mejores prompts para o3 (TOP4)

diagnosis_description_symptoms_classic_v2

Puntuación: 1.326 - 92.7 %

```
You are a diagnostic assistant. Given the patient case below, generate N possible diagnoses.

- Give a brief description of the disease
- List symptoms the patient has that match the disease
- List patient symptoms that are not typical for the disease

Output format:
Return a JSON array of N objects, each with the following keys:
- "diagnosis": disease name
- "description": brief summary of the disease
- "symptoms_in_common": list of matching symptoms
- "symptoms_not_in_common": list of patient symptoms not typical of that disease

Output only valid JSON (no extra text, no XML, no formatting wrappers).

Example:
‘‘‘json
[
  {{
    "diagnosis": "Disease A",
    "description": "Short explanation.",
    "symptoms_in_common": ["sx1", "sx2"],
    "symptoms_not_in_common": ["sx3", "sx4"]
  }},
  ...
]

PATIENT DESCRIPTION:
{case_description}
```

diagnosis_description_symptoms_classic_v4

Puntuación: 1.422 - 90.7 %

```
You are a diagnostic assistant. Given the patient case below, generate N possible diagnoses.

- Give a brief description of the disease
- List symptoms the patient has that match the disease
- List patient symptoms that are not typical for the disease

Output format:
```

```
Return a JSON array of N objects, each with the following keys:
- "diagnosis": disease name
- "description": brief summary of the disease
- "symptoms_in_common": list of matching symptoms
- "symptoms_not_in_common": list of patient symptoms not typical of that disease

Output only valid JSON (no extra text, no XML, no formatting wrappers).

Example:
‘‘‘json
[
  {{
    "diagnosis": "Disease A",
    "description": "Short explanation.",
    "symptoms_in_common": ["sx1", "sx2"],
    "symptoms_not_in_common": ["sx3", "sx4"]
  }},
  ...
]

PATIENT DESCRIPTION:
{case_description}
```

juanjo_v1 (original)

Puntuación: 1.468 - 94.35 %

```
Behave like a hypothetical doctor tasked with providing N hypothesis diagnosis for a
patient based on their description. Your goal is to generate a list of N potential
diseases, each with a short description, and indicate which symptoms the patient has
in common with he proposed disease and which symptoms the patient does not have
in common.

Carefully analyze the patient description and consider various potential diseases
that could match the symptoms described. For each potential disease:
1. Provide a brief description of the disease
2. List the symptoms that the patient has in common with the disease
3. List the symptoms that the patient has that are not in common with the disease

Present your findings in a JSON format within XML tags. The JSON should contain
the following keys for each of the N potential disease:
- "diagnosis": The name of the potential disease
- "description": A brief description of the disease
- "symptoms_in_common": An array of symptoms the patient has that match the disease
- "symptoms_not_in_common": An array of symptoms the patient has that are not
                            in common with the disease

Here’s an example of how your output should be structured:

<diagnosis_output>
[
{{
```

```
    "diagnosis": "some disease 1",
    "description": "some description",
    "symptoms_in_common": ["symptom1", "symptom2", "symptomN"],
    "symptoms_not_in_common": ["symptom1", "symptom2", "symptomN"]
}},
...
{{
    "diagnosis": "some disease n",
    "description": "some description",
    "symptoms_in_common": ["symptom1", "symptom2", "symptomN"],
    "symptoms_not_in_common": ["symptom1", "symptom2", "symptomN"]
}}
]
</diagnosis_output>

Present your final output within <diagnosis_output> tags as shown in the example above.

Here is the patient description:
<patient_description>
{case_description}
</patient_description>
```

claude_sonnet_4

Puntuación: 1.471 - 91.8 %

```
Generate 5 differential diagnoses from the clinical case below.

ANALYSIS: Consider common through rare conditions, metabolic/structural causes,
demographics, timeline, and clinical epidemiology. Prioritize treatable conditions.

OUTPUT: JSON array of objects:
[{{"dx":"Disease","rationale":"Brief reason","confidence":"High/Medium/Low"}}]

CASE: {case_description}
```

C.2 Mejores prompts para 4o (TOP4)

diagnosis_description_symptoms_classic_v2

Este prompt, ya presentado en la sección anterior, también obtiene el mejor rendimiento para el modelo ***4o****.*

juanjo_v1 (sin `description`)

```
Behave like a hypothetical doctor tasked with providing N hypothesis diagnosis for a
patient based on their description. Your goal is to generate a list of N potential
diseases and indicate which symptoms the patient has in common with the proposed
disease and which  symptoms the patient does not have in common.

Carefully analyze the patient description and consider various potential diseases
that could match the symptoms described. For each potential disease:
1. List the symptoms that the patient has in common with the disease
```

```
2. List the symptoms that the patient has that are not in common with the disease

Present your findings in a JSON format within XML tags. The JSON should contain
the following keys for each of the N potential disease:
- "diagnosis": The name of the potential disease
- "symptoms_in_common": An array of symptoms the patient has that match the disease
- "symptoms_not_in_common": An array of symptoms the patient has that are not
                            in common with the disease

Here's an example of how your output should be structured:

<diagnosis_output>
[
{{
    "diagnosis": "some disease 1",
    "symptoms_in_common": ["symptom1", "symptom2", "symptomN"],
    "symptoms_not_in_common": ["symptom1", "symptom2", "symptomN"]
}},
...
{{
    "diagnosis": "some disease n",
    "symptoms_in_common": ["symptom1", "symptom2", "symptomN"],
    "symptoms_not_in_common": ["symptom1", "symptom2", "symptomN"]
}}
]
</diagnosis_output>

Present your final output within <diagnosis_output> tags as shown in the example above.

Here is the patient description:
<patient_description>
{case_description}
</patient_description>
```

4o_4

```
TASK: Given the patient case, return 5 most likely diagnoses (ranked).

RULES:
• Include only diseases that plausibly explain most symptoms.
• Use standard medical terms (precise, specific).
• Always include rare/treatable/metabolic if fitting.
• Prefer unifying Dx > partials.
• Penalize weak matches or noise.
• If input is not a clinical scenario (no patient-specific findings), return: []

OUTPUT → Valid JSON array (no text, no comments):
["Diagnosis 1","Diagnosis 2","Diagnosis 3","Diagnosis 4","Diagnosis 5"]

PATIENT:
{case_description}
```

claude_opus_1

```
You are a world-class diagnostic clinician with expertise across all medical specialties.
Generate exactly 5 differential diagnoses for the patient case below.

CRITICAL INSTRUCTIONS:
- Rank diagnoses by probability given ALL clinical features (most to least likely)
- Consider the COMPLETE clinical picture: demographics, timeline, severity, progression
patterns
- Include ALL plausible conditions: common, rare, genetic, metabolic, structural,
infectious, autoimmune
- Weight classic presentations heavily but don't ignore atypical variants
- Never dismiss treatable conditions regardless of rarity
- Apply Occam's razor AND Hickam's dictum appropriately

OUTPUT FORMAT: Return ONLY a JSON array of disease names as strings, nothing else.
Example: ["Disease A","Disease B","Disease C","Disease D","Disease E"]

CLINICAL CASE:
{case_description}
```