Router RL para selección de LLM

Alumno: Iñaki Larrumbide

Profesor: Miguel Augusto Azar

Materia: Aprendizaje por refuerzo

1 · Introducción

Hoy se ven cada vez más los productos basados en LLM y arquitecturas multi-agente, capaces de resolver tareas desde muy simples hasta altamente complejas. Esto plantea un nuevo desafío: **elegir el modelo óptimo para cada consulta**. Cada LLM ofrece capacidades distintas y existe un claro *trade-off* entre potencia (calidad) y latencia / coste.

Modelo	Ventana tokens	Latencia base	Coste relativo
GPT-3.5-16 k	16 k	Baja	Bajo
Gemini-32 k	32 k	Media	Medio
GPT-4o-128 k	128 k	Alta	Alto

El reto es elegir, para cada consulta, el modelo que maximice **calidad** sin disparar **latencia** ni coste.

En lugar de reglas fijas proponemos un **agente de Aprendizaje por Refuerzo (RL) tabular** que aprende esa política a partir de recompensas simuladas.

2 · Modelado RL

Componente	Definición
Acciones (3)	0 → GPT-3.5-16 k · 1 → GPT-4o-128 k · 2 → Gemini-32 k
Estados (36)	Triple tupla <**context_window**, **n_tools**, **resp_size**> • *context_window* ∈ {**small** (< 4 k), **medium** (4-32 k), **large** (> 32 k)} • *n_tools* ∈ {0, 1, 2, ≥ 3} • *resp_size* ∈ {**short** (< 256 tok), **medium** (256-1024), **long** (> 1024)}
Recompensa	$R = quality - \lambda \cdot latency\_s$ con λ = 1.0. Si *context* = large y el modelo no soporta la ventana (3.5-16 k o Gemini-32 k) ⇒ **R = –10**
Transición	Episodio = 100 consultas; se descarta el estado *(large, 0 tools, short)* por ser irreal.

3 · Parámetros de simulación

- **Base (μ ± σ) de calidad y latencia** por (n_tools, resp_size, modelo) extraída de benchmarks internos y datos ficticios para una primera version.

- Penalizaciones/bonificaciones añadidas:

    - *context* = *medium*: 3.5 –0.10 q +0.20 s · 4o +0.15 q +0.20 s.

    - *context* = *large*: 3.5 –0.30 q +0.40 s · 4o +0.25 q +0.40 s · Gemini –0.05 q +0.25 s.

    - *n_tools* ≥ *3* **o** *resp_size* = *long*: 4o +0.15 q extra.

---

4 · Algoritmos implementados

Método	Hiper-parámetros	Episodios para converger
Q-Learning tabular	α = 0.1, γ = 0.95, ε-greedy (1 → 0.05)	≈ 40 k
SARSA(0)	mismos α, γ, ε	convergencia más suave, valor final menor

---

5 · Resultados finales

Entrenamiento: 120 000 episodios · $\lambda = 1.0$

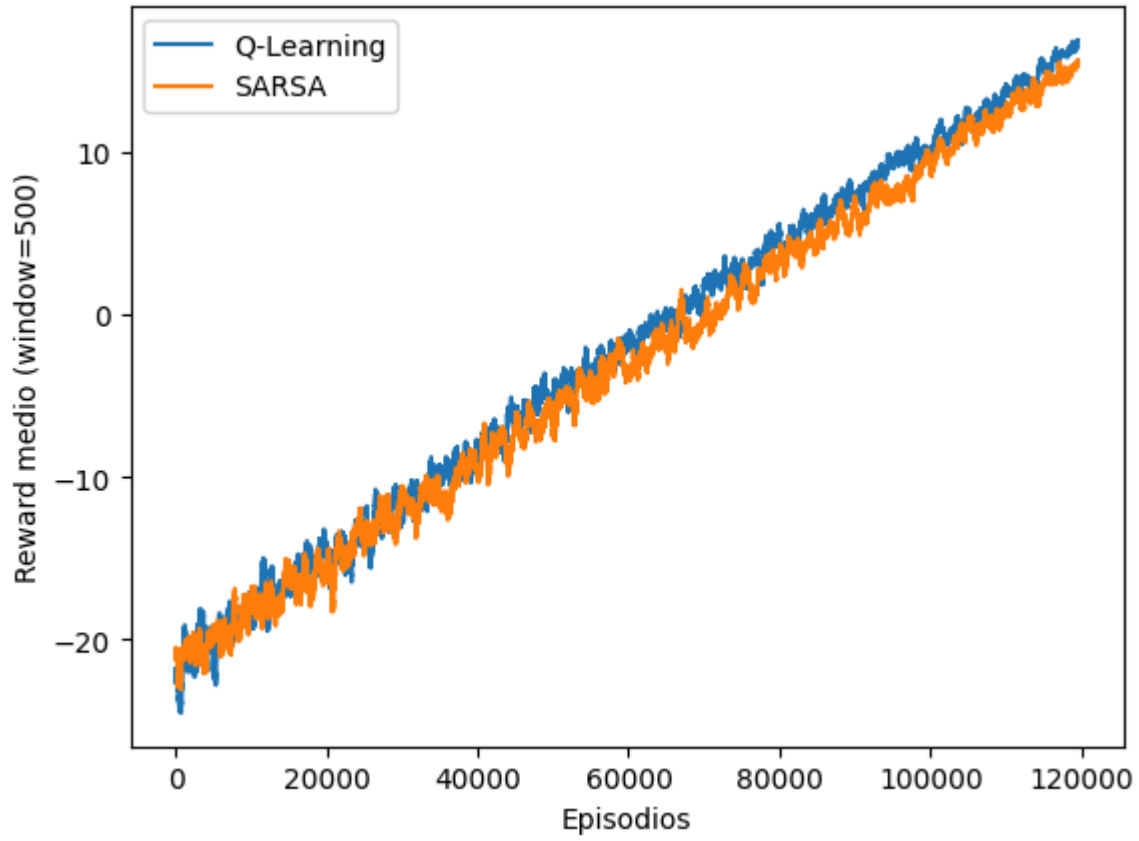
Política	**Reward medio (10 000 ep.)**
Q-Learning	**+19.33**
SARSA(0)	+15.29
Baseline (siempre GPT-3.5-16 k)	–38.99

5.1 Política aprendida (extracto)

context / tools / resp	**Modelo elegido**
small · 0 · short	3.5-16 k
small · 0 · medium / long	Gemini-32 k
small · 1 · long	GPT-4o-128 k

small · ≥ 3 · short	GPT-4o-128 k
medium · 1 · medium / long	GPT-4o-128 k
medium · 2 · long	GPT-4o-128 k
large · (≥ 1) · *	**GPT-4o-128 k** (único factible)

5.2 Convergencia



Q-Learning → plateau ≈ +19; SARSA → +15. La RL supera al baseline por ≈ +59 pts.

---

6 · Discusión

- **Decisiones lógicas**
    - Consultas simples → 3.5-16 k.
    - Prompts medianos con ≤ 2 tools → Gemini-32 k.
    - Contexto grande / ≥ 3 tools / respuesta larga → GPT-4o-128 k.
- **Ganancia neta** — El agente RL aprende a usar **GPT-3.5-16 k o Gemini-32 k** en todos los casos donde la calidad adicional de GPT-4o no compensa su coste y su mayor latencia.

  Así:
    - **Ahorro directo**: menos llamadas a GPT-4o ⇒ menor factura de tokens.
    - **Respuesta más rápida**: consultas simples pasan de ~700 ms (4o) a ~250 ms (3.5).
    - **Calidad preservada**: en los estados "pesados" (contexto grande, ≥3 tools o respuesta larga) sí se selecciona GPT-4o, manteniendo la precisión donde realmente importa.

El resultado agregado se ve en el reward: de **–38.99** (baseline que siempre usa 3.5) pasamos a **+19.33** con la política RL, evidenciando que el agente equilibra correctamente **coste, latencia y calidad**.

- **Robustez** — con $\lambda \in [0.8, 1.2]$ la política conserva la misma estructura; solo cambian algunos estados limítrofes.

---

7 · Conclusiones

- El agente **Q-Learning tabular** aprende una política alineada a la lógica de negocio.
- Reduce costes (menos uso de GPT-4o) manteniendo calidad cuando hace falta.
- Implementable en producción como diccionario de 36 × 3 valores → decisión < 1 ms.
- Con feedback real puede evolucionar a *online learning* incremental.
