ChatGPT

Paso 4 – Política de Verificación y Gating Antialucinación

1. Resumen ejecutivo

Para garantizar que las respuestas del agente sean fiables, es necesario integrar **mecanismos de verificación** que detecten y corrijan alucinaciones o errores, sin sacrificar demasiado la fluidez de interacción. Basamos la política en tres componentes complementarios:

1. **Chain of Verification (CoVe):** técnica de *multi-etapa* que hace que el modelo cree un borrador, formule preguntas de verificación, responda esas preguntas de manera aislada y finalmente sintetice una respuesta verificada. Los experimentos muestran que CoVe reduce significativamente las alucinaciones en tareas de QA y generación [1].
2. **Dynamic Self-Verify Decoding (DSVD):** estrategia de generación que verifica continuamente la salida y, cuando detecta errores, aplica un mecanismo de *rollback* para corregirlos. DSVD integra una arquitectura de auto-verificación paralela y un mecanismo de retroceso dinámico, mejorando la veracidad y exactitud factual sin grandes sacrificios de eficiencia [2].
3. **Gating adaptativo (RAS):** el agente debe decidir *cuándo* aplicar CoVe o DSVD en función del riesgo percibido (p. ej. presencia de afirmaciones factuales, contradicciones, nivel de entropía) y de los recursos disponibles (presupuesto de tokens/latencia). Este gating forma parte del RAS (Real–Simbólico–Imaginario) descrito en los pasos anteriores.

2. Componentes de verificación

2.1 Chain of Verification (CoVe)

El método CoVe sigue cuatro pasos: (i) **borrador inicial** de la respuesta, (ii) **planificación de preguntas de verificación** para comprobar hechos y pasos de razonamiento, (iii) **resolución independiente** de cada pregunta para evitar sesgos, y (iv) **síntesis** en una respuesta final [1]. Este pipeline reduce la propagación de errores y ha demostrado disminuir alucinaciones en una variedad de tareas [1].

Ventajas:

- Reduce las alucinaciones en contextos donde hay preguntas factuales o cadenas de razonamiento largas.
- Permite la incorporación de fuentes externas y verificadores simbólicos.

Limitaciones:

- Añade latencia proporcional al número de preguntas generadas.
- Para tareas muy largas o generación de texto libre, puede ser costoso.

2.2 Dynamic Self-Verify Decoding (DSVD)

DSVD se centra en la **verificación en tiempo real** durante la generación. El modelo mantiene una **arquitectura de auto-verificación paralela** para evaluar la calidad de cada token generado y un

mecanismo de retroceso dinámico que permite regresar a estados previos cuando se detecta una alucinación; tras el retroceso, la generación continúa con un re-muestreo penalizado [2]. Los autores demuestran que DSVD incrementa la veracidad en varios benchmarks y que su sobrecoste es moderado [2].

Ventajas:

- Detecta errores locales tan pronto como aparecen; corrige sobre la marcha.
- La arquitectura paralela y el retroceso local permiten mantener la latencia bajo control [2].

Limitaciones:

- Necesita acceso a las probabilidades internas del modelo y a su estado latente; no siempre disponible en entornos cerrados.
- Más adecuada para generación continua (chat/decoding) que para respuestas breves o verificaciones post-hoc.

2.3 Prompt Risk Mitigation (PRM)

A falta de una definición estandarizada de "PRM" en la literatura, adoptamos este término para englobar **políticas de mitigación de riesgo en el prompting**. PRM consiste en emplear instrucciones e inyecciones de contexto para restringir la generación (por ejemplo, pedir siempre fuentes, usar plantillas de respuesta controladas y activar verificadores simbólicos). Este componente se apoya en reglas simbólicas y no requiere acceder al estado interno del modelo. Se alinea con el nivel **Simbólico** del RAS.

Ventajas:

- Sencillo de implementar: funciona a nivel de prompt y no exige modificar el decodificador del modelo.
- Puede combinarse con CoVe y DSVD (por ejemplo, instruyendo al modelo a aplicar CoVe cuando detecte un dato factual).

Limitaciones:

- No detecta alucinaciones durante la generación, sólo reduce la probabilidad de que ocurran.
- La efectividad depende de la obediencia del modelo a las instrucciones.

3. Política de gating y decisión

El RAS debe decidir, en cada turno, si activa un mecanismo de verificación y cuál. Proponemos una función `VerifierPolicyGate` con los siguientes inputs: `risk_flags`, `evidence_available`, `mode`, `budget` (tokens/latencia) y `rsi_state` (Real/Simbólico/Imaginario). La decisión devuelve un plan de verificación de entre: `NONE`, `PRM_ONLY`, `COVE_LIGHT`, `COVE_FULL`, `DSVD_STREAM`.

3.1 Criterios de activación

- **Factual claims sin evidencia:** si la salida contiene afirmaciones factuales no sustentadas (`risk_flags.factual_claims=true` y `evidence_available=false`), activar CoVe en modo ligero (`COVE_LIGHT`): generar preguntas de verificación sólo para las afirmaciones

críticas. Esto se basa en la evidencia de que CoVe reduce las alucinaciones al obligar al modelo a verificar hechos [1].

- **Contradicciones o errores lógicos:** si se detectan contradicciones internas (`risk_flags.contradiction=true`), aplicar `COVE_FULL` para descomponer la cadena de razonamiento y verificar cada paso.

- **Generación larga y dinámica (chat continuo):** si el modelo está generando textos prolongados (por ejemplo resúmenes largos o código extenso), aplicar `DSVD_STREAM` para supervisar en tiempo real y corregir mediante rollback. La DSVD ha demostrado mejorar la exactitud factual durante la generación continua sin sacrificar practicidad [2].

- **Presupuesto y latencia:** si el `budget_state` está en el modo crítico (latencia alta, tokens bajos), limitar la verificación a `PRM_ONLY` o incluso `NONE`, para no exceder recursos.

- **Entropía baja y sin riesgo:** si la entropía del enrutamiento es baja y no hay flags de riesgo, omitir verificaciones (`NONE`).

3.2 Niveles de verificación

1. **NONE:** no se realiza verificación; se usa sólo cuando hay bajo riesgo y el resultado es trivial.
2. **PRM_ONLY:** se aplica mitigación de riesgo en el prompt (plantillas con peticiones de fuentes, instrucciones de verificación manual). Útil cuando no se puede acceder a CoVe o DSVD y el presupuesto es bajo.
3. **COVE_LIGHT:** versión simplificada de CoVe; se generan pocas preguntas clave de verificación y se responden de manera aislada antes de sintetizar la respuesta.
4. **COVE_FULL:** CoVe completo; se descompone todo el razonamiento en pasos y se verifica cada uno. Recomendado para contenidos críticos (por ejemplo, respuestas legales o médicas).
5. **DSVD_STREAM:** se activa durante la generación de texto en streaming, implementando la auto-verificación paralela y el rollback dinámico [2].

4. Evaluación y métricas

Para medir la efectividad de la política de verificación se proponen las siguientes métricas:

1. **Tasa de alucinaciones:** proporción de respuestas con afirmaciones incorrectas antes y después de aplicar cada mecanismo. La literatura demuestra que DSVD mejora la veracidad y precisión en cinco benchmarks [2] y que CoVe disminuye alucinaciones en varias tareas [1].
2. **Latencia promedio:** tiempo adicional introducido por cada mecanismo (CoVe, DSVD, PRM). DSVD tiene un coste moderado gracias a su verificación paralela [2], mientras que CoVe puede aumentar la latencia proporcionalmente al número de preguntas.
3. **Uso de tokens/recursos:** cantidad de tokens consumidos y llamadas a herramientas adicionales durante la verificación.
4. **Satisfacción del usuario:** medida cualitativa (por ejemplo calificación de la respuesta) para evaluar si la mejora en fidelidad compensa el tiempo de espera.

5. Riesgos y mitigaciones

1. **Dependencia de parámetros internos:** DSVD requiere acceso a la arquitectura interna y a los logits del modelo; si el entorno no lo permite, se debe recurrir a CoVe o PRM.

2. **Sobrecarga de verificación:** aplicar CoVe completo o DSVD en exceso puede provocar latencias inaceptables; el gating debe priorizar las situaciones de riesgo alto.
3. **Falsos negativos:** un mecanismo de entropía mal calibrado podría no disparar la verificación cuando es necesaria; se recomienda ajustar los umbrales sobre datos reales.
4. **Plantillas y obediencia:** PRM depende de que el modelo siga instrucciones; en algunos casos el modelo puede ignorarlas. Incluir verificación simbólica complementaria puede mitigar este riesgo.

6. Próximos pasos

1. **Implementar el** `VerifierPolicyGate` como un módulo independiente que reciba los flags de riesgo y devuelva un plan de verificación.
2. **Ajustar umbrales** de activación (entropía, presupuesto, severidad de las afirmaciones) con datos reales y feedback del usuario.
3. **Diseñar evaluaciones A/B** comparando distintos mecanismos (PRM, CoVe, DSVD) en tus tareas habituales, midiendo alucinación, latencia y satisfacción.

---

[1] [2309.11495] Chain-of-Verification Reduces Hallucination in Large Language Models
https://arxiv.org/abs/2309.11495

[2] DSVD: Dynamic Self-Verify Decoding for Faithful Generation in Large Language Models
https://arxiv.org/html/2503.03149v1