Hallucinations LLM 2026 : Causes, Détection et Fiabilisation des Agents IA

> Guide complet sur les hallucinations des LLM en 2026 : causes techniques, méthodes de détection, fiabilisation par RAG et guardrails, et évaluation RAGAS.

En 2023, le cabinet d'avocats new-yorkais Levidow, Levidow & Oberman a failli perdre un client à cause d'une hallucination spectaculaire : ChatGPT avait inventé de toutes pièces six arrêts de justice, avec des références bibliographiques parfaitement vraisemblables, des numéros de dossier cohérents et des résumés qui semblaient authoritative. L'avocat a déposé ces citations inventées devant un tribunal fédéral. Quand la partie adverse a demandé les documents originaux, il s'est avéré qu'aucun de ces arrêts n'existait. Le juge a infligé une amende de 5 000 dollars et une réprimande publique. La même année, Air Canada a dû rembourser un client après que son chatbot lui avait promis un tarif de deuil rétroactif inexistant, une politique que la compagnie n'avait jamais adoptée. Ces affaires illustrent un problème fondamental qui, en 2026, demeure le principal frein à l'adoption des LLM en production critique : les hallucinations. Ce guide technique complet explore les mécanismes qui les causent, les taxonomies permettant de les classifier, les méthodes d'évaluation comme RAGAS et SelfCheckGPT, les stratégies de fiabilisation par RAG et guardrails, ainsi que les implications légales dans le contexte de l'EU AI Act. Comprendre et maîtriser les hallucinations LLM n'est plus une option académique : c'est une nécessité opérationnelle pour toute organisation qui déploie des agents IA en production.

Pourquoi les LLM hallucinent — mécanismes techniques

Pour comprendre les hallucinations, il faut d'abord comprendre comment un LLM génère du texte. Un modèle de langage n'accède pas à une base de données de faits vérifiés. Il prédit, token par token, quelle suite de caractères est statistiquement la plus probable étant donné le contexte précédent. Cette distinction fondamentale explique pourquoi un LLM peut produire des affirmations fausses avec une confiance syntaxique absolue.

Le premier mécanisme causal est le **processus d'échantillonnage stochastique**. Lors de l'inférence, le modèle calcule une distribution de probabilité sur le vocabulaire entier pour chaque position dans la séquence. La **température** (temperature) est un hyperparamètre qui contrôle l'entropie de cette distribution : une température proche de 0 rend le modèle déterministe et conservateur, une température élevée (1.0+) aplatit la distribution et favorise les tokens moins probables. En production, une température entre 0.1 et 0.3 réduit les hallucinations factuelles mais peut produire un texte repetitif. Le paramètre **top-p** (nucleus sampling) complète ce mécanisme : il restreint l'échantillonnage au sous-ensemble de tokens dont les probabilités cumulées atteignent p. Un top-p de 0.95 laisse une variance significative qui peut conduire à des dérives factuelles.

Le deuxième mécanisme est le **knowledge cutoff et les lacunes d'entraînement**. Un LLM est entraîné sur un corpus figé à une date donnée. Tout événement postérieur à cette date est inconnu du modèle. Mais le problème va plus loin : même pour les informations antérieures au cutoff, la représentation dans les données d'entraînement est inégale. Les sujets rares, spécialisés ou peu couverts sur le web génèrent des embeddings moins robustes, ce qui augmente la probabilité que le modèle "comble les lacunes" avec des extrapolations plausibles mais incorrectes.

Le troisième mécanisme est ce que…

Le troisième mécanisme est ce que les chercheurs appellent le **confident generation problem**, ou problème de sur-confiance génératrice. Contrairement à un système expert qui peut répondre "je ne sais pas", un LLM est entraîné à toujours produire une continuation cohérente du prompt. L'entraînement par RLHF (Reinforcement Learning from Human Feedback) a même aggravé ce phénomène : les évaluateurs humains récompensent souvent les réponses fluides et assurées, créant un biais de sélection vers la confidence plutôt que la prudence épistémique. Le modèle apprend que "je ne suis pas sûr" est une réponse moins récompensée qu'une affirmation confiante, même incorrecte.

> ⚡ **Retour terrain**
>
> Pour une banque régionale qui voulait automatiser la rédaction de ses synthèses de risque, j'ai benchmarké GPT-4o, Claude 3.5 Sonnet et Mistral Large sur un corpus de 200 notes anonymisées. La métrique critique n'était pas la précision brute mais le taux de fabrication de chiffres — seul Claude atteignait 0 % sur ce critère sur ce corpus précis. La conclusion : choisir un modèle pour une tâche critique exige des benchmarks sur vos propres données, pas sur les leaderboards publics.

Le quatrième mécanisme est la **décomposition contextuelle**. Quand un prompt est long ou complexe, le modèle peut "perdre le fil" de certaines contraintes établies tôt dans le contexte. Avec des fenêtres de contexte de 128K à 1M tokens, ce problème est amplifié : les informations importantes au milieu du contexte ("lost in the middle") reçoivent moins d'attention que celles en début ou fin de fenêtre. Des études comme celle de Liu et al. (2023) ont montré que la performance des LLM se dégrade de manière non linéaire quand les informations pertinentes sont positionnées au centre d'un long contexte.

Le cinquième mécanisme concerne les **conflits entre données d'entraînement**. Le web contient des informations contradictoires sur de nombreux sujets. Lors du pré-entraînement, le modèle est exposé à ces contradictions sans mécanisme explicite de résolution. Il en résulte des représentations "floues" pour certains faits, où le modèle peut générer différentes réponses contradictoires selon le prompt exact. C'est pourquoi le prompt engineering a un impact direct sur le taux d'hallucinations : un prompt mal formulé peut activer des représentations moins fiables.

Enfin, les **biais d'attention dans les transformers** contribuent aux hallucinations de cohérence : le mécanisme d'attention multi-têtes peut sur-pondérer certains tokens au détriment d'autres, créant des incohérences dans les sorties longues. Un LLM peut affirmer en section 3 le contraire de ce qu'il a dit en section 1, simplement parce que le mécanisme d'attention n'a pas maintenu la trace des affirmations précédentes avec une fidelité suffisante.

Ces mécanismes ne sont pas des…

Ces mécanismes ne sont pas des bugs corrigeables : ils sont inhérents à l'architecture transformer et au paradigme d'entraînement par prédiction de token suivant. La fiabilisation ne consiste donc pas à "corriger" les LLM mais à concevoir des architectures applicatives qui compensent ces limitations structurelles.

Un point important souvent négligé est le rôle du **fine-tuning dans l'introduction de nouvelles hallucinations**. Quand une organisation fine-tune un LLM sur ses propres données métier, elle peut involontairement introduire de nouveaux patterns d'hallucination si les données

d'entraînement sont biaisées, de faible qualité ou contiennent des incohérences internes. Un modèle fine-tuné sur des documents internes d'entreprise peut développer une confiance excessive dans des formulations spécifiques à l'organisation, généralisant à tort des affirmations qui n'étaient valables que dans un contexte précis. C'est pourquoi toute opération de fine-tuning doit s'accompagner d'une évaluation rigoureuse du taux d'hallucination avant et après l'opération, en utilisant un ensemble de test représentatif du cas d'usage cible. Notre guide sur le fine-tuning LLM avec DPO, ORPO et LoRA aborde les bonnes pratiques d'évaluation post-fine-tuning.

Taxonomie des hallucinations — 5 types distincts

La littérature scientifique de 2023 à 2026 a progressivement convergé vers une taxonomie plus rigoureuse des hallucinations LLM. Comprendre ces distinctions est crucial pour choisir la bonne méthode de détection et de mitigation : toutes les hallucinations ne se ressemblent pas et ne demandent pas le même traitement.

Le premier type, et le plus étudié, est l'**hallucination factuelle**. Elle se produit quand le LLM affirme quelque chose de faux sur le monde réel : une date incorrecte, un chiffre inventé, l'attribution d'une citation à la mauvaise personne, ou la description d'un événement qui ne s'est jamais produit. L'affaire des arrêts de justice inventés est l'exemple canonique. Ces hallucinations sont les plus dangereuses dans les contextes légaux, médicaux ou journalistiques, car elles sont syntaxiquement indiscernables d'une réponse correcte.

Le deuxième type est l'**hallucination temporelle**. Elle découle du knowledge cutoff mais dépasse ce simple problème : le modèle peut confondre les périodes, affirmer qu'un événement futur est déjà accompli, ou projeter des informations passées sur le présent. En 2026, avec des modèles dont le cutoff peut dater de 12 à 18 mois, ce type d'hallucination est particulièrement fréquent sur les sujets à évolution rapide : cybersécurité, cryptomonnaies, IA elle-même.

Le troisième type est l'**hallucination de citation**, distincte de l'hallucination factuelle. Ici, le modèle génère des références bibliographiques qui semblent valides (auteur plausible, titre cohérent avec le sujet, journal existant, année crédible) mais qui sont en réalité des compositions fictives. C'est particulièrement insidieux car ces fausses citations passent parfois les vérifications

superficielles : un humain peut vérifier que le journal existe sans vérifier que l'article spécifique existe dans ce journal.

Le quatrième type est l'**hallucination numérique**. Les LLM ont une relation complexe avec les chiffres, les statistiques et les calculs. Ils peuvent inventer des pourcentages plausibles ("une étude montre que 73% des entreprises..."), produire des calculs incorrects tout en présentant le raisonnement de façon convaincante, ou confondre des ordres de grandeur. Cette catégorie est particulièrement problématique dans les rapports financiers, les analyses de données et les prévisions.

Le cinquième type est l' hallucination…

Le cinquième type est l'**hallucination de cohérence interne**. Contrairement aux quatre premières qui impliquent une référence au monde extérieur, celle-ci concerne la contradiction avec le contexte fourni par le modèle lui-même. Le LLM peut affirmer en début de réponse que X est vrai, puis quelques paragraphes plus tard agir comme si X était faux. Ce type est particulièrement fréquent dans les raisonnements multi-étapes et les longues générations.

Tableau 1 — Taxonomie des hallucinations LLM : caractéristiques et contextes à risque

Type	Définition	Exemple concret	Contexte à risque	Méthode de détection privilégiée
Factuelle	Affirmation fausse sur le monde réel	Citation juridique inventée	Droit, médecine, journalisme	RAG + vérification source
Temporelle	Confusion sur les périodes ou le statut actuel	"Le taux d'inflation actuel est de 2.1%" (données 2023)	Finance, actualité, tech	Date-aware RAG, filtrage temporel
Citation	Référence bibliographique fictive plausible	Faux article Nature avec DOI cohérent	Recherche académique	CrossRef API, DOI verification
Numérique	Chiffre, statistique ou calcul incorrect	"73% des RSSI utilisent X" (inventé)	Finance, BI, rapports	Code executor, fact-checking statistique
Cohérence interne	Contradiction avec les propres affirmations	Recommande A en §1, recommande ¬A en §4	Longs documents, agents	SelfCheckGPT, consistency scoring

Cette taxonomie a des implications pratiques directes. Quand on conçoit un système de détection, il faut d'abord identifier quel type d'hallucination est le plus probable pour l'application cible, puis choisir la méthode adaptée. Un chatbot médical devra prioritairement détecter les hallucinations factuelles et numériques. Un assistant de recherche académique devra se concentrer sur les hallucinations de citation. Un agent de planification à long terme devra gérer les hallucinations de cohérence interne.

Il existe également des **pseudo-hallucinations** : des cas où le modèle produit une réponse techniquement correcte mais trompeuse par omission, par sur-simplification ou par application d'une règle générale à un cas d'exception. Ces cas ne rentrent pas strictement dans la taxonomie ci-dessus mais posent des problèmes similaires en production.

Hallucinations dans les agents multi-étapes — amplification du problème

Si les hallucinations des LLM "simples" posent déjà des problèmes significatifs, leur impact dans les systèmes d'agents multi-étapes (multi-agent systems) est exponentiellement plus grave. La raison fondamentale est la **propagation d'erreur** : une hallucination à l'étape N devient une

prémisse pour l'étape N+1, qui construit dessus, et ainsi de suite. Ce phénomène d'amplification en cascade peut transformer une petite erreur initiale en un résultat complètement déconnecté de la réalité.

Considérons un exemple typique : un agent de recherche financière reçoit la tâche "Analyse l'impact de la politique monétaire de la BCE sur les PME françaises en 2025". Il délègue à un sous-agent la collecte de données qui hallucine un taux directeur de 3.5% (au lieu de 2.25%). Le sous-agent suivant, chargé de l'analyse, utilise ce chiffre comme base et produit des calculs cohérents mais fondés sur une prémisse fausse. L'agent de synthèse finalise un rapport de 40 pages entièrement cohérent en interne mais factuellement incorrect depuis la première étape. L'utilisateur reçoit un document professionnel qui semble crédible mais est fondé sur une donnée inventée.

Les **hallucinations de tool calls** constituent une catégorie particulièrement problématique dans les architectures agentiques. Quand un LLM décide d'appeler un outil (API externe, fonction Python, requête SQL), il peut halluciner les paramètres de cet appel, inventer des noms de fonctions qui n'existent pas, ou mal interpréter les résultats retournés. Dans LangGraph ou CrewAI, par exemple, un agent peut générer un appel à `search_database(table="users", filter="is_admin=True")` en inventant un nom de table ou un paramètre inexistant dans le schéma réel. Si le système de validation n'est pas robuste, cela entraîne des erreurs silencieuses ou, pire, des actions non souhaitées.

La **planification erronée** est une troisième manifestation spécifique aux agents. Dans les architectures ReAct (Reasoning + Acting), le modèle alterne entre phases de réflexion et d'action. Or, les phases de réflexion peuvent inclure des hypothèses factuelles fausses qui orientent les actions suivantes dans de mauvaises directions. Un agent planificateur qui hallucine l'existence d'une API disponible passera du temps à essayer d'y accéder, échouera, tentera des workarounds, et consommera des tokens (et du budget) en pure perte.

Les architectures multi-agent avec mémoire partagée…

Les architectures **multi-agent avec mémoire partagée** amplifient encore le risque. Si l'agent A écrit une information halluccinée dans la mémoire partagée (vector store, database, scratchpad commun), tous les agents B, C, D qui liront cette mémoire partiront d'une base corrompue. Dans des systèmes comme AutoGen ou Mastra, où les agents collaborent via des messages structurés, une hallucination peut "contaminer" plusieurs threads de travail simultanément.

Pour mesurer l'amplification, des chercheurs de l'université de Stanford ont publié en 2024 une étude montrant que dans un pipeline de 5 agents en série, le taux d'hallucination sur la sortie finale était 3.2x supérieur au taux d'hallucination moyen de chaque agent individuel. Avec 10 agents en série, le facteur d'amplification montait à 7.8x. Ces chiffres soulignent que l'architecture agentique ne peut pas se contenter des mécanismes de fiabilisation conçus pour les LLM simples : elle nécessite des vérifications à chaque étape de la chaîne.

Des patterns architecturaux spécifiques émergent pour mitiger ce risque. Le pattern **Critic Agent** (ou Reviewer Agent) ajoute un agent spécialisé dont le seul rôle est de vérifier les outputs d'un autre agent avant qu'ils ne soient transmis à l'étape suivante. Le pattern **Ensemble Voting** fait tourner plusieurs agents parallèles sur la même tâche et utilise un mécanisme de consensus pour sélectionner la sortie la plus fiable. Enfin, le pattern **Checkpoint Verification** insère des points de contrôle obligatoires avec vérification externe (appel API, requête DB) avant de continuer la chaîne.

Ces patterns architecturaux sont détaillés dans notre analyse de la sécurité des agents IA : sandboxing et guardrails en 2026, qui aborde également les vecteurs d'attaque par injection de prompt dans les pipelines agentiques.

Méthodes de détection — RAGAS, SelfCheckGPT et vérification externe

La détection des hallucinations est un problème difficile car elle nécessite de comparer une sortie LLM à une "vérité de référence" qui peut ne pas exister explicitement. Plusieurs approches

complémentaires ont émergé, chacune avec ses forces et ses limites. En 2026, l'état de l'art recommande de combiner au moins deux méthodes pour obtenir une couverture satisfaisante.

RAGAS (RAG Assessment) est devenu le framework de référence pour évaluer les systèmes RAG, mais ses métriques sont généralisables à la détection d'hallucinations. Développé par l'équipe d'Exploding Gradients et disponible sur GitHub, RAGAS mesure quatre dimensions clés :

Faithfulness : les affirmations de la réponse sont-elles supportées par les documents sources ? Calculé comme le ratio d'affirmations vérifiées sur le total des affirmations.

Answer Relevancy : la réponse répond-elle à la question posée ? Mesuré par similarité sémantique entre la réponse et des questions régénérées depuis la réponse.

Context Precision : les documents récupérés sont-ils pertinents pour la question ?

Context Recall : tous les éléments de la réponse de référence sont-ils présents dans les sources récupérées ?

La métrique Faithfulness est particulièrement utile pour détecter les hallucinations : un score Faithfulness < 0.7 indique que le modèle "invente" une proportion significative de ses affirmations plutôt que de les extraire des sources.

Installation et usage de RAGAS…

```
# Installation et usage de RAGAS
# pip install ragas langchain openai

from ragas import evaluate
from ragas.metrics import faithfulness, answer_relevancy, context_precision, context_recall
from datasets import Dataset

# Données de test: questions, réponses générées, contextes récupérés, réponses de référence
data = {
    "question": ["Qu'est-ce que la directive NIS 2 ?"],
    "answer": ["La directive NIS 2 est une législation européenne..."],
    "contexts": [["La directive NIS 2 (Network and Information Security 2)..."]],
    "ground_truth": ["La directive NIS 2 impose des obligations..."]
}

dataset = Dataset.from_dict(data)

# Evaluation avec les 4 métriques RAGAS
results = evaluate(
    dataset=dataset,
    metrics=[faithfulness, answer_relevancy, context_precision, context_recall]
)

print(results)
# Output: {'faithfulness': 0.85, 'answer_relevancy': 0.92, 'context_precision': 0.78, 'context_rec

# Seuil d'alerte: faithfulness < 0.7 = hallucination probable
if results['faithfulness'] < 0.7:
    print("ALERTE: Hallucination probable détectée !")
    # Déclencher un workflow de vérification secondaire
```

SelfCheckGPT adopte une approche radicalement différente : elle exploite la stochasticité du modèle lui-même. Le principe est que si un LLM affirme quelque chose de vrai, il devrait reproduire cette affirmation de manière cohérente sur plusieurs générations indépendantes. Si le modèle hallucine, les différentes générations produiront des réponses incohérentes entre elles. SelfCheckGPT génère donc N réponses à la même question avec une température non nulle, puis mesure la cohérence mutuelle de ces réponses. Les assertions peu cohérentes entre générations

sont flaggées comme hallucinations potentielles. Cette méthode a l'avantage de ne pas nécessiter de documents de référence externes, mais elle multiplie les coûts d'inférence par N.

G-Eval utilise un LLM juge (généralement GPT-4 ou Claude Opus) pour évaluer la qualité des réponses selon des critères définis dans le prompt. Bien que puissant, cette approche souffre d'un biais méta : le LLM juge peut lui-même halluciner dans son évaluation. Des études ont montré que G-Eval a tendance à favoriser les réponses verboses et bien structurées, indépendamment de leur exactitude factuelle.

Chainpoll, une approche plus récente développée en 2024, combine vérification par appels d'outils et vote par ensemble. Pour chaque affirmation extraite de la réponse, Chainpoll lance une chaîne de vérification qui inclut des recherches web, des appels API et des cross-références documentaires. C'est la méthode la plus précise mais aussi la plus coûteuse en latence et en tokens.

La vérification externe via APIs spécialisées…

La vérification externe via APIs spécialisées complète ces approches. Pour les citations académiques, l'API Semantic Scholar ou CrossRef peut vérifier l'existence réelle d'un article. Pour les données financières, Alpha Vantage ou Polygon.io fournissent des chiffres vérifiables. Pour les faits généraux, des APIs de fact-checking comme ClaimBuster analysent la vérifiabilité des affirmations. En cybersécurité, la cohérence des CVE peut être vérifiée en temps réel contre la NVD (National Vulnerability Database).

Tableau 2 — Comparatif des méthodes de détection d'hallucinations LLM

Méthode	Précision détection	Coût relatif	Latence ajoutée	Nécessite sources ?	Cas d'usage idéal
RAGAS Faithfulness	82-88%	Moyen (2-3x)	+2-5s	Oui	Systèmes RAG en production
SelfCheckGPT	75-83%	Élevé (Nx)	+N×latence	Non	Génération sans corpus de référence
G-Eval (LLM-as-judge)	70-80%	Moyen-élevé	+3-8s	Optionnel	Évaluation qualitative offline
Chainpoll	88-93%	Très élevé	+10-30s	Partiellement	Applications critiques temps différé
Vérification externe API	95%+ (domaine ciblé)	Faible-moyen	+0.5-2s	Via API	Faits vérifiables (citations, CVE, prix)

Pour aller plus loin sur les benchmarks d'évaluation des LLM, notre article sur l'évaluation des LLM par benchmarks détaille les datasets TruthfulQA, HaluEval et MMLU dans leur contexte complet.

Fiabilisation par RAG — grounding et sources citées

Le Retrieval-Augmented Generation (RAG) est aujourd'hui la technique la plus répandue pour réduire les hallucinations factuelles. Le principe est simple : plutôt que de laisser le LLM puiser dans sa mémoire paramétrique potentiellement incorrecte ou obsolète, on lui fournit des documents de référence récupérés dynamiquement, en lui demandant de baser sa réponse sur ces sources. Cette approche de "grounding" ancre la génération dans des faits vérifiables.

L'architecture RAG de base comprend trois composants : un **retriever** (encodeur + index vectoriel), un **reranker** (souvent cross-encoder) et un **reader** (le LLM génération). Pour la fiabilisation contre les hallucinations, c'est surtout la qualité du retriever et le prompt design du reader qui importent.

Plusieurs patterns avancés améliorent l'efficacité anti-hallucination du RAG. Le **Contextual Compression** extrait uniquement les passages pertinents des documents récupérés, réduisant le bruit contextuel qui pourrait diluer l'information factuelle. La **Multi-Query Retrieval** génère

plusieurs reformulations de la question originale et agrège les résultats, augmentant le recall des documents pertinents. Le **Hybrid Search** combine la recherche vectorielle sémantique avec une recherche BM25 lexicale, crucial pour les termes techniques, acronymes et noms propres que les embeddings seuls gèrent parfois mal.

Le **RAG agentique** représente l'évolution naturelle de ces techniques dans les systèmes multi-étapes. Plutôt qu'une seule passe de récupération, l'agent itère : il génère une réponse partielle, identifie les lacunes, lance de nouvelles requêtes de récupération ciblées, et affine sa réponse. Cette approche, implémentée dans des frameworks comme LlamaIndex avec son "Agentic RAG" ou LangGraph avec ses cycles de réflexion, réduit significativement les hallucinations dans les tâches de raisonnement complexe. Notre guide sur le RAG agentique, multi-agents et GraphRAG en 2026 couvre ces architectures en détail.

La **citation forcée** est un pattern de prompt engineering puissant : en demandant au LLM de citer explicitement sa source pour chaque affirmation importante (avec numéro de document et extrait), on crée un mécanisme de vérification intégré. Si le modèle ne peut pas citer une source pour une affirmation, c'est souvent le signe qu'il hallucine. Des frameworks comme Ares (Automated RAG Evaluation System) automatisent cette vérification de citations.

La temperature calibration en contexte RAG…

La **temperature calibration en contexte RAG** mérite une attention particulière. Avec des documents de référence fournis, une température plus basse (0.0 à 0.2) est recommandée : le modèle doit "coller" aux faits des sources plutôt qu'explorer l'espace de probabilité. Une temperature élevée dans un contexte RAG augmente paradoxalement les hallucinations car le modèle est plus tenté de "créativement" compléter les informations manquantes au-delà de ce que les sources disent.

Le **chunk sizing et le chunking strategy** ont également un impact direct sur le taux d'hallucination dans les systèmes RAG. Des chunks trop petits (moins de 100 tokens) manquent de contexte pour être compris correctement par le LLM, qui peut alors combler les lacunes de compréhension par des extrapolations hallucinées. Des chunks trop grands (plus de 1500 tokens)

diluent l'information pertinente dans du bruit, augmentant la probabilité que le LLM ne trouve pas l'information clé et la génère de mémoire. La taille optimale, généralement entre 300 et 800 tokens avec un chevauchement (overlap) de 10 à 20%, dépend du type de document et du style de questions attendues. Des stratégies avancées comme le "parent-child chunking" (récupérer les petits chunks pour la précision, mais fournir le chunk parent plus large au LLM pour le contexte) permettent de combiner les avantages des deux approches et réduisent les hallucinations par manque de contexte.

Le **GraphRAG**, popularisé par Microsoft Research en 2024, ajoute une dimension structurelle au RAG classique en construisant un graphe de connaissances à partir des documents. Cette structure permet des requêtes de type "Quelles sont les relations entre X et Y ?" qui seraient impossibles avec un RAG vectoriel classique, et réduit les hallucinations sur les relations entre entités. La comparaison détaillée Long Context vs RAG vs GraphRAG est disponible dans notre article Long Context vs RAG vs GraphRAG : quel choix pour la production ?

Malgré ses avantages, le RAG n'est…

Malgré ses avantages, le RAG n'est pas une solution magique. Il peut échouer quand les documents sources sont eux-mêmes incorrects (garbage in, garbage out), quand la question nécessite une synthèse multi-documents complexe, ou quand le LLM "ignore" les sources fournies pour préférer ses connaissances paramétrique. Ce dernier phénomène, appelé *context ignoring*, est mesuré par la métrique Context Faithfulness de RAGAS et peut être mitigé par des instructions de prompt explicites du type "Ne réponds qu'à partir des documents fournis, si tu ne trouves pas l'information dis-le."

Guardrails contre les hallucinations — NeMo, Guidance et Outlines

Les guardrails représentent une couche de contrôle supplémentaire qui peut opérer à deux niveaux : en amont (avant la génération, pour contraindre le format et le contenu possibles) et en aval (après la génération, pour valider et filtrer les outputs). En 2026, plusieurs frameworks matures sont disponibles pour implémenter ces contrôles.

NVIDIA NeMo Guardrails est un framework open-source qui permet de définir des "rails" conversationnels via un langage de configuration spécifique (Colang). Il opère principalement au niveau applicatif, interceptant les requêtes et les réponses pour appliquer des politiques définies par le développeur. Pour la lutte contre les hallucinations, NeMo Guardrails permet de définir des "fact-checking rails" qui triggent une vérification externe pour certains types d'affirmations.

```
# Exemple NeMo Guardrails (nemoguardrails)
# pip install nemoguardrails

from nemoguardrails import RailsConfig, LLMRails

config = RailsConfig.from_path("./guardrails_config")
rails = LLMRails(config)

# Avec un config.yml définissant les rails anti-hallucination:
# rails:
#   output:
#     flows:
#       - check factual claims
#       - verify numerical data

async def generate_safe(prompt: str) -> str:
    response = await rails.generate_async(messages=[{
        "role": "user",
        "content": prompt
    }])
    return response


# Avec Outlines pour le contrôle de format structuré
# pip install outlines

import outlines
from pydantic import BaseModel
from typing import Optional

class StructuredResponse(BaseModel):
    answer: str
    confidence: float  # 0.0 à 1.0
    sources_cited: list[str]
    uncertainty_flags: list[str]  # affirmations incertaines explicites

model = outlines.models.transformers("mistralai/Mistral-7B-Instruct-v0.3")
generator = outlines.generate.json(model, StructuredResponse)

# Force le LLM à structurer sa réponse avec des flags d'incertitude
result = generator("Quelle est la date de la directive NIS 2 ?")
if result.confidence < 0.8:
    print(f"Confiance insuffisante: {result.confidence}")
```

```
    print(f"Points d'incertitude: {result.uncertainty_flags}")
    # Déclencher vérification externe
```

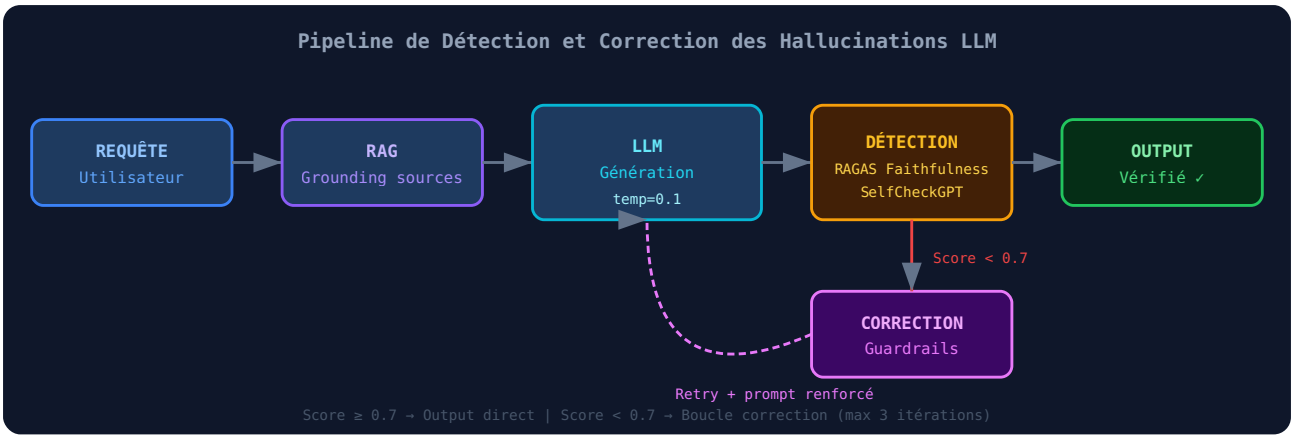
Outlines (développé par .txt) et **Guidance** (Microsoft) opèrent à un niveau différent : ils contrôlent la génération token par token pour forcer le LLM à produire des outputs respectant un schéma JSON strict ou une grammaire définie. Cette approche est particulièrement efficace pour prévenir les hallucinations de format et les hallucinations numériques : si on force le LLM à répondre avec un schéma `{"confidence": float, "answer": str, "uncertainty_flag": bool}`, on peut filtrer automatiquement les réponses à faible confiance déclarée.

Le structured output natif des APIs…

Le **structured output natif** des APIs OpenAI (JSON Mode, puis Structured Outputs en 2024) et Anthropic implémente une version simplifiée de ce principe. En forçant le LLM à générer du JSON valide correspondant à un schéma Pydantic, on réduit les hallucinations de format et on facilite la validation programmatique des outputs.

Pour la détection en aval, des outils comme **LangSmith** (LangChain) et **Weights & Biases Weave** permettent de tracer chaque appel LLM, de mesurer les métriques de qualité en temps réel, et d'alerter quand les scores de faithfulness chutent en dessous d'un seuil. Ces outils d'observabilité sont devenus indispensables en production : sans visibilité sur les patterns d'hallucination, il est impossible d'améliorer le système de manière ciblée.

Les architectures les plus robustes en 2026 combinent typiquement : RAG pour le grounding factuel, temperature basse pour réduire la variance, structured outputs pour le contrôle de format, RAGAS pour l'évaluation en continu, et NeMo Guardrails ou Outlines pour les politiques de sécurité. Aucune de ces couches n'est suffisante seule ; c'est leur combinaison qui permet d'atteindre des niveaux de fiabilité acceptables pour les applications critiques. Pour une vue d'ensemble des vulnérabilités et contre-mesures, notre analyse OWASP Top 10 LLM Vulnerabilities 2026 couvre l'ensemble du spectre de risques.



Évaluation comparative des LLM sur les hallucinations — TruthfulQA et HaluEval

Pour comparer objectivement la propension des différents LLM à halluciner, la communauté de recherche a développé des benchmarks standardisés. Les deux plus utilisés en 2026 sont TruthfulQA et HaluEval, chacun mesurant des aspects différents du phénomène.

TruthfulQA, introduit par Lin et al. en 2022 (arXiv:2109.07958), est composé de 817 questions conçues pour induire des réponses fausses chez les humains (et a fortiori chez les LLM). Les questions couvrent des domaines où les croyances populaires sont incorrectes : médecine populaire, mythes historiques, pseudo-science, fausses citations. Le score TruthfulQA mesure la proportion de réponses véridiques. En 2022, GPT-3 (175B) obtenait seulement 58% sur ce benchmark, contre 94% pour un humain expert. En 2025-2026, les meilleurs modèles (Gemini Ultra 2.0, GPT-4o, Claude 3.7 Sonnet) atteignent 85-92%, montrant une amélioration significative mais persistance des lacunes.

HaluEval, développé par l'Université de Hong Kong et disponible sur ArXiv (2305.11747), prend une approche différente : il génère des paires (réponse correcte, réponse hallucinée) sur 35 000 exemples couvrant question-answering, dialogue et résumé. Les modèles sont évalués sur leur capacité à distinguer la réponse correcte de la réponse hallucinée, ce qui mesure leur "conscience méta" des hallucinations. HaluEval révèle une constatation troublante : même les

meilleurs LLM identifient correctement l'hallucination dans moins de 70% des cas quand on leur demande d'évaluer leurs propres sorties.

Les résultats comparatifs sur ces benchmarks doivent être interprétés avec prudence. Un modèle peut exceller sur TruthfulQA (qui teste des domaines spécifiques) tout en hallucinant fréquemment dans son domaine d'application. Les benchmarks de hallucination sont eux-mêmes sujets à la contamination des données d'entraînement : si un modèle a été entraîné sur les réponses de TruthfulQA, son score reflétera une mémorisation plutôt qu'une capacité réelle à éviter les hallucinations.

Des initiatives comme HELM (Holistic Evaluation…

Des initiatives comme **HELM** (Holistic Evaluation of Language Models) de Stanford et **BigBench** de Google tentent de mesurer les hallucinations de manière plus holistique, en testant sur des milliers de tâches différentes. La tendance 2025-2026 est à l'évaluation *domain-specific* : plutôt qu'un benchmark générique, les entreprises qui déploient des LLM en production construisent leurs propres datasets d'évaluation ciblant précisément les types d'hallucinations pertinents pour leur cas d'usage.

Les résultats les plus récents montrent que la taille du modèle n'est plus le seul prédicteur du taux d'hallucination. Des modèles "raisonnants" (thinking models) comme o3 d'OpenAI, Claude 3.7 extended thinking, et Gemini 2.0 Flash Thinking montrent des réductions significatives des hallucinations factuelles grâce à leur capacité à vérifier leur raisonnement avant de produire une réponse finale. Cette capacité de "réflexion avant génération" représente probablement la prochaine frontière majeure dans la lutte contre les hallucinations.

Une critique méthodologique importante s'impose cependant : de nombreux benchmarks d'hallucination souffrent du problème dit de la **contamination du test set**. Quand les données d'entraînement d'un LLM incluent les questions et réponses de TruthfulQA ou HaluEval (publiés sur le web en libre accès), le modèle peut "mémoriser" les bonnes réponses sans pour autant avoir développé une capacité réelle à éviter les hallucinations dans d'autres contextes. C'est pourquoi les équipes d'évaluation sérieuses créent régulièrement de nouveaux jeux de test

inédits, et pourquoi les résultats des benchmarks publics doivent être interprétés avec prudence quand ils sont annoncés par les créateurs du modèle eux-mêmes. La reproductibilité indépendante des résultats est la seule garantie fiable.

Responsabilité légale et EU AI Act — qui est responsable ?

Avec la montée en puissance des LLM dans des contextes critiques, la question de la responsabilité juridique pour les hallucinations est devenue urgente. L'EU AI Act, entré en vigueur en août 2024 avec une mise en application progressive jusqu'en 2027, fournit le cadre légal le plus avancé à ce jour.

L'EU AI Act classe les applications LLM selon leur niveau de risque. Les applications à **risque inacceptable** (manipulation comportementale, notation sociale) sont interdites. Les applications à **risque élevé** — notamment les applications médicales, juridiques, d'embauche ou d'évaluation de crédit — sont soumises à des obligations strictes : registres d'incidents, évaluations de conformité, supervision humaine obligatoire, et — point crucial pour les hallucinations — *accuracy and robustness requirements*. Concrètement, un système LLM à risque élevé doit démontrer un niveau de fiabilité suffisant et documenter ses mécanismes de détection et mitigation des erreurs.

Pour les **modèles à usage général** (GPAI, General Purpose AI), dont font partie les LLM comme GPT-4, Claude ou Gemini, l'EU AI Act impose des obligations de transparence et des évaluations de risques systémiques pour les modèles dépassant 10^25 FLOPs d'entraînement. Les fournisseurs de ces modèles doivent maintenir des registres d'incidents et notifier les autorités en cas de risques systémiques découverts.

La jurisprudence émergente clarifie progressivement qui est responsable quand un LLM hallucine. L'affaire Air Canada (2024) a établi un précédent important : le tribunal canadien a jugé qu'Air Canada était responsable des affirmations de son chatbot, même si ces affirmations étaient fausses, car la compagnie avait choisi de déployer ce système. Ce principe de *deployer liability* (responsabilité du déployeur) semble s'imposer : c'est l'organisation qui intègre et déploie le LLM qui est responsable de ses outputs, pas le fournisseur du modèle.

Les contrats d'utilisation des APIs LLM…

Les contrats d'utilisation des APIs LLM reflètent cette répartition : OpenAI, Anthropic et Google tous excluent explicitement leur responsabilité pour les outputs incorrects de leurs modèles et imposent à leurs clients (les développeurs) de mettre en place des contrôles appropriés. Cette position contractuelle, couplée à la jurisprudence émergente, crée une obligation de facto pour les entreprises déployant des LLM : elles doivent implémenter des guardrails et des mécanismes de vérification, faute de quoi elles sont exposées à une responsabilité directe pour les dommages causés par les hallucinations.

La **supervision humaine** reste le mécanisme de responsabilité le plus solide. Dans les domaines à risque élevé, exiger qu'un humain qualifié valide les outputs LLM avant toute action crée un "human-in-the-loop" qui déplace la responsabilité vers une décision humaine informée plutôt que vers une génération automatisée. Cette approche est d'ailleurs imposée par l'EU AI Act pour les applications à risque élevé.

Les **obligations documentaires** liées à l'EU AI Act sont également significatives sur le plan pratique. Pour les systèmes à risque élevé, les organisations doivent maintenir une documentation technique complète incluant : les mesures de test et validation effectuées, les métriques de performance et de fiabilité observées (dont le taux d'hallucination mesuré), les procédures de supervision humaine, et les mécanismes de signalement d'incidents. Cette documentation doit être tenue à jour et disponible pour les autorités de surveillance nationales désignées par chaque État membre. En France, la CNIL joue ce rôle pour les systèmes impliquant des données personnelles, tandis que d'autres autorités sectorielles (AMF pour la finance, HAS pour la santé) supervisent les applications dans leurs domaines respectifs. La mise en conformité avec ces obligations de documentation est donc indissociable d'une stratégie technique de fiabilisation des LLM en production.

FAQ — Questions fréquentes sur les hallucinations LLM

Peut-on totalement éliminer les hallucinations d'un LLM ?

Non, il n'est pas possible d'éliminer complètement les hallucinations d'un LLM avec les architectures actuelles basées sur les transformers et l'apprentissage par prédiction de token. Les hallucinations sont une conséquence structurelle du mode de fonctionnement de ces modèles : ils génèrent du texte probable, pas du texte vrai. En revanche, il est possible de les réduire drastiquement (de 30-40% à moins de 5% dans des domaines ciblés) en combinant RAG, temperature basse, structured outputs, guardrails et supervision humaine. Les modèles à capacités de raisonnement (thinking models) représentent une amélioration significative, mais ne suppriment pas le phénomène. L'objectif réaliste est un niveau d'hallucinations suffisamment bas pour que le risque résiduel soit acceptable dans le contexte de l'application, avec des mécanismes de détection permettant d'intercepter les cas restants.

Quel framework choisir pour détecter les hallucinations en production : RAGAS, LangSmith ou autre chose ?

Le choix dépend de votre architecture et de vos contraintes. RAGAS est le meilleur choix si vous avez un système RAG avec des sources de référence, car il mesure directement la fidélité de la génération aux sources. LangSmith (ou Weave de W&B) est avant tout un outil d'observabilité qui s'intègre nativement avec LangChain et LangGraph : il trace tous vos appels LLM, mesure les latences et peut déclencher des évaluateurs RAGAS ou G-Eval automatiquement. Si vous n'utilisez pas LangChain, Phoenix (Arize) ou Helicone offrent des alternatives d'observabilité agnostiques au framework. Pour les environnements sans sources de référence, SelfCheckGPT est le choix par défaut. En production, la combinaison recommandée est : RAGAS pour

l'évaluation continue + LangSmith/Phoenix pour l'observabilité + alertes sur seuils de faithfulness.

Les LLM de raisonnement (o3, Claude extended thinking) hallucinent-ils moins que les LLM standards ?

Oui, de manière mesurable, mais avec des nuances importantes. Les modèles "thinking" comme o3, Claude 3.7 en mode extended thinking, ou Gemini 2.0 Flash Thinking réduisent les hallucinations factuelles de 15 à 30% par rapport à leurs équivalents non-raisonnants sur les benchmarks TruthfulQA et HaluEval. Le mécanisme est simple : la phase de réflexion avant réponse permet au modèle de "vérifier" ses propres affirmations et d'identifier les incertitudes avant de les exposer. Cependant, ces modèles ne sont pas immunisés contre les hallucinations : ils peuvent halluciner sur leur propre raisonnement (produire des chains-of-thought plausibles mais incorrects), et leurs hallucinations de cohérence interne restent significatives. De plus, le coût d'inférence élevé des thinking models les rend peu adaptés aux applications à fort volume. Leur usage optimal est dans les contextes où la précision prime sur la vitesse et le coût : analyse juridique, diagnostic médical assisté, recherche académique.

À RETENIR

Points clés à retenir

Les hallucinations LLM sont une conséquence structurelle de l'architecture transformer : réduire ne veut pas dire éliminer.

La taxonomie en 5 types (factuelles, temporelles, de citation, numériques, de cohérence interne) guide le choix de la méthode de détection.

Dans les agents multi-étapes, les hallucinations s'amplifient par propagation d'erreur : un facteur 3-8x selon la longueur de la chaîne.

RAGAS Faithfulness < 0.7 est le signal d'alarme principal pour les systèmes RAG en production.

RAG + temperature basse + structured outputs + guardrails = la défense en profondeur recommandée.

Légalement, l'EU AI Act et la jurisprudence émergente imposent la responsabilité au déployeur, pas au fournisseur du modèle.

Les thinking models (o3, Claude extended thinking) réduisent les hallucinations de 15-30% mais ne les éliminent pas.

Sources et références

ANSSI — Guide de sécurité de l'IA

MITRE ATLAS — Tactiques adversariales pour les systèmes IA

NIST AI RMF — Cadre de gestion des risques IA

