Détection et Prévention des Jailbreaks dans les Modèles de Langage à Grande Échelle (LLM)

Auteur : Prénom NOM
Affiliation : Nom de l'institution
Email : adresse@mail.com

1er juin 2025

Résumé

Les modèles de langage de grande taille (LLMs) ont récemment démontré des performances remarquables sur une large gamme de tâches. Toutefois, leur puissance les rend vulnérables aux attaques de type jailbreak, où des utilisateurs malveillants tentent de contourner les mesures de sécurité afin d'obtenir des réponses interdites ou dangereuses. Dans cet article, nous présentons une architecture hybride de défense combinant un modèle détecteur de jailbreak entraîné et un modèle juge génératif, chargé d'évaluer la nature malveillante d'un prompt ou d'une réponse. Lorsqu'un prompt utilisateur ne déclenche pas d'alerte au niveau du détecteur, la réponse générée est ensuite analysée par le juge pour déterminer si elle constitue une violation effective. Nous enrichissons cette architecture par l'utilisation de plusieurs modèles LLMs de test, permettant une évaluation comparative de leur robustesse face aux attaques. Notre infrastructure s'appuie notamment sur les modèles LLaMA 3.2, mistral et phi, avec un tableau de bord intégré pour visualiser la répartition des interactions bénignes ou malveillantes selon le modèle . Les résultats expérimentaux montrent que l'approche combinée permet de detecter et bloquer plus de 67% des cas de tentatives de jailbreak, offrant ainsi une base solide et extensible pour renforcer la sécurité des LLMs dans des scénarios réels.

1 Introduction

Les LLMs (Large Language Models), tels que GPT, LLaMA, Claude ou Zephyr, ont connu une adoption fulgurante ces dernières années, transformant de nombreux secteurs grâce à leur capacité à comprendre et à générer du langage naturel avec une grande précision. Leur polyvalence les rend particulièrement utiles dans des domaines variés, allant de l'assistance conversationnelle à la rédaction de code, en passant par la traduction automatique ou la génération de contenu.

Cependant, cette puissance n'est pas sans risques. Les LLMs sont également vulnérables à une classe croissante d'attaques appelées *jailbreaks*. Ces attaques consistent à concevoir des invites (prompts) malicieuses permettant de contourner les mécanismes de sécurité intégrés aux modèles afin d'obtenir des réponses potentiellement dangereuses ou interdites, telles que la fabrication d'armes, des discours haineux, ou des conseils illégaux. Malgré l'introduction de filtres de sécurité basés sur des règles, l'apprentissage par renforcement (RLHF), ou des techniques de filtrage en post-traitement, les attaquants trouvent régulièrement de nouvelles méthodes pour détourner le comportement des modèles.

Face à ces limitations, nous proposons dans cet article une architecture hybride, combinant apprentissage automatique supervisé et raisonnement génératif, afin de renforcer la détection et la prévention des comportements malveillants. Notre système repose sur un détecteur spécialisé, entraîné pour classifier les prompts suspects avant exécution, couplé à un modèle *juge* qui évalue la réponse produite par un modèle cible pour en déterminer la légitimité ou la dangerosité. Cette double vérification permet de réduire les faux négatifs tout en limitant les faux positifs.

De plus, afin d'évaluer la robustesse de cette architecture dans des contextes variés, nous introduisons l'utilisation de plusieurs modèles LLMs de test, comme LLaMA 3.2 mistral et phi, dans le but d'analyser les différences de comportement face à des tentatives similaires de contournement. Cette diversité offre une perspective comparative essentielle pour caractériser les vulnérabilités spécifiques à chaque modèle.

L'implémentation technique repose sur une infrastructure modulaire avec journalisation centralisée et tableau de bord interactif, permettant une analyse approfondie des interactions. Les statistiques générées incluent des métriques fines selon le modèle ciblé, le taux de détection initial et le verdict final du juge. Les résultats expérimentaux confirment l'efficacité de notre approche, avec une réduction significative des jailbreaks non détectés, tout en conservant un faible taux de faux negatifs.

Dans la suite de cet article, nous présentons l'état de l'art relatif aux attaques jailbreaks et aux défenses existantes, puis nous décrivons notre méthodologie, les modèles utilisés, les scénarios de test, et les résultats obtenus. Nous concluons par une discussion sur les limites actuelles et les pistes futures, notamment en matière d'évaluation de la robustesse des LLMs en environnement hostile.

2 Travaux connexes

La sécurité des modèles de langage de grande taille (LLMs) a récemment attiré une attention croissante de la communauté scientifique. Plusieurs recherches se sont concentrées sur la compréhension des vulnérabilités des LLMs face aux attaques de type jailbreak, ainsi que sur le développement de méthodes de détection et de mitigation.

2.1 Détection des prompts malveillants

Zou et al. (2023) ont introduit *PromptGuard*, une approche utilisant l'apprentissage automatique pour détecter les prompts conçus pour contourner les restrictions. Leur système repose sur l'extraction de caractéristiques linguistiques et sémantiques, couplées à un classificateur supervisé. De manière similaire, Chen et al. (2022) ont proposé une méthode de détection basée sur l'analyse syntaxique et des patterns suspects dans les instructions utilisateurs.

Cependant, ces approches se concentrent majoritairement sur le contenu du prompt, sans considérer le résultat produit par le modèle, ce qui limite leur efficacité face à des prompts ambigus ou adversariaux bien masqués.

2.2 Détection post-réponse

Pour pallier cette limitation, certaines recherches ont proposé d'évaluer la réponse générée. Notamment, le projet *LLM Safety Gym* (OpenAI, 2023) teste systématiquement les réponses des modèles à des prompts sensibles, en les comparant à une base de données de comportements attendus. De même, la méthode *DecodingTrust* (2023) combine des tests automatisés avec une analyse humaine pour juger si une réponse enfreint les normes éthiques.

Notre architecture reprend cette logique avec l'introduction d'un modèle *juge* autonome, permettant d'évaluer dynamiquement la dangerosité de chaque réponse, même lorsque le prompt semble bénin.

2.3 Techniques d'adversarial prompting

L'étude de Zou et al. (2023) sur les attaques *AutoDAN* a mis en évidence la capacité des attaquants à générer automatiquement des prompts d'évasion efficaces en utilisant des algorithmes

d'optimisation. Wang et al. (2023), avec leur framework *JailbreakBench*, ont fourni une évaluation standardisée de plusieurs modèles face à des attaques réalistes, montrant des taux de réussite alarmants même pour les modèles les plus récents.

Ces travaux motivent l'utilisation de plusieurs modèles de test dans notre architecture, afin d'évaluer la robustesse des défenses face à une diversité d'attaques et de comportements modèles.

2.4 Approches hybrides

Enfin, quelques initiatives explorent des architectures hybrides. Xu et al. (2023) ont proposé une chaîne de filtres incluant un détecteur initial suivi d'une étape de validation par un autre modèle. Bien que conceptuellement proche de notre proposition, leur méthode n'intègre pas d'analyse comparative multi-modèles, ni de tableau de bord permettant une observation dynamique.

2.5 Résumé

En résumé, bien que plusieurs travaux aient abordé la sécurité des LLMs, peu d'approches combinent la détection proactive des prompts et l'analyse critique des réponses dans une architecture unifiée. Notre travail se distingue par l'intégration de ces deux dimensions, tout en introduisant une dimension multi-modèles pour enrichir l'évaluation comparative et renforcer la résilience du système.

3 Méthodologie

Dans cette section, nous détaillons l'architecture proposée, les composants clés du pipeline de défense, les modèles utilisés, ainsi que le processus décisionnel permettant de gérer les prompts suspects ou malveillants. L'objectif principal est d'intégrer une détection précoce des attaques tout en conservant une capacité à juger a posteriori les contenus générés.

3.1 Architecture générale

Notre pipeline de sécurité est structuré en trois modules principaux : un **détecteur** de jailbreaks, un ou plusieurs **modèles de test** chargés de générer la réponse à l'utilisateur, et un **modèle juge** responsable de l'évaluation de la réponse produite. Le flux d'une interaction typique se déroule comme suit :

1. **Soumission du prompt :** L'utilisateur envoie un prompt au système.
2. **Analyse par le détecteur :** Un modèle spécialisé, entraîné à détecter les prompts malveillants, évalue le texte soumis.
3. **Cas 1 — Prompt détecté comme dangereux :** La requête est bloquée et l'utilisateur reçoit un message d'avertissement.
4. **Cas 2 — Prompt jugé bénin :** Le prompt est envoyé à un ou plusieurs **modèles de test (LLMs)** pour génération.
5. **Analyse de la réponse par le juge :** La réponse produite est transmise à un modèle juge, distinct, chargé de détecter si le contenu constitue une violation (réponse de type jailbreak). Si c'est le cas, le prompt est bloque et est ajoute comme un log a un fichier .csv qui sera utilise pour ameliorer notre modele de detection.
6. **Validation finale :** Si la réponse est jugée sûre, elle est transmise à l'utilisateur. Sinon, une alerte est enregistrée, et l'utilisateur reçoit un message neutre.

Cette architecture assure une double protection, agissant à la fois en amont (sur le prompt) et en aval (sur la réponse), réduisant ainsi significativement le risque de contournement.

3.2 Modèle détecteur

Le modèle détecteur est un classificateur entraîné à partir d'un jeu de données constitué de prompts bénins et de tentatives de jailbreak. L'entraînement est basé sur une représentation vectorielle TF-IDF des prompts, suivie d'une classification par SVM . Le modèle est optimisé pour minimiser les faux négatifs tout en conservant un taux de faux positifs acceptable.

3.3 Modèles de test

Les modèles de test sont des LLMs utilisés pour simuler une interaction normale avec un assistant IA. Ils jouent un rôle clé dans l'évaluation de la robustesse du système face à différentes familles de modèles. Dans notre infrastructure, nous utilisons :

— **LLaMA 3.2** — modèle performant et représentatif des LLMs de nouvelle génération.
— **Mistral 7B** — modèle open source plus léger, utile pour évaluer des réponses alternatives.
— **Phi-2** — modèle compact avec comportement distinct.

Chaque modèle est sollicité individuellement selon le scénario, ce qui permet une analyse comparative de leur vulnérabilité.

3.4 Modèle juge

Le modèle développé est basé sur l'architecture RoBERTa-base (Robustly optimized BERT approach), un modèle de langage pré-entraîné de type transformeur qui a montré d'excellentes performances pour les tâches de classification de texte. Le modèle a été adapté pour la classification binaire avec 2 labels (benign vs jailbreak) en ajoutant une couche de classification finale.

3.5 Infrastructure technique

L'ensemble des composants est orchestré par un backend Django. Les communications entre les modules sont réalisées via des appels REST internes. Les modèles sont hébergés localement via `Ollama`, garantissant le contrôle total des flux et une latence minimale. Un tableau de bord intégré permet la visualisation en temps réel des statistiques suivantes :

— Proportion de prompts bénins vs. malveillants.
— Répartition des réponses benignes bloquées selon le modèle.

3.6 Flux décisionnel global

Le diagramme de la Figure 1 présente le pipeline complet du système et la logique de décision entre les modules.

4 Expérimentations et Résultats

Dans cette section, nous présentons le protocole expérimental, les jeux de données utilisés, les métriques d'évaluation retenues, ainsi que les résultats obtenus lors des différentes phases de test de notre système hybride de détection et de jugement des attaques de type jailbreak. L'objectif est d'évaluer la robustesse de l'architecture proposée, sa capacité à intercepter les tentatives d'exploitation, et la complémentarité entre les différents modules.

4.1 Jeux de données

Nos expérimentations se sont appuyes principalement sur :

— **218 Prompts jailbreak (malveillants)** : Generés par des modeles comme*DeepSeek*, *ChatGpt*. Ces prompts visent explicitement à contourner les garde-fous des LLMs.

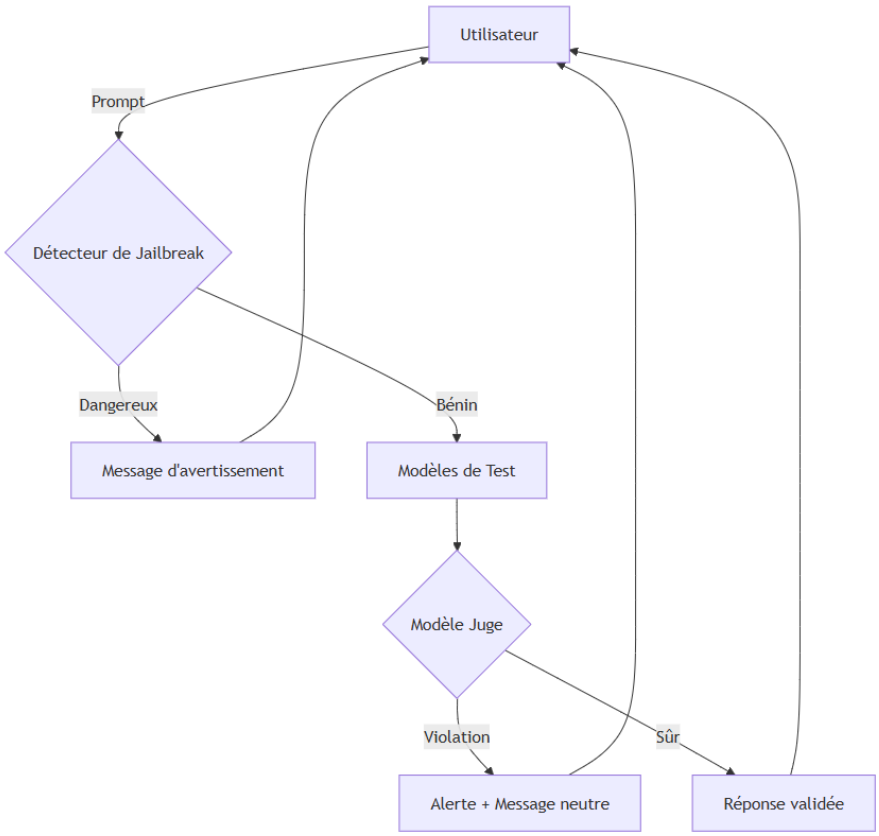

FIGURE 1 – Architecture hybride de défense contre les jailbreaks dans les LLMs

4.2 Protocoles de test

Pour évaluer la performance de notre système, nous avons envoye les 218 prompts a chaque modele de test et nous avons stocke toutes les classifications dans la base de donnees. Chaque interaction respecte le processus decrit ci-dessus.

4.3 Métriques d'évaluation

Pour chaque composant, nous avons mesuré :

Détecteur :

— **Taux de détection (TPR)** : proportion de prompts malveillants correctement détectés.

Juge :

— **P**roportion de prompts malveillants detectes par le juge.

Système global (détecteur + juge) :

— **Taux de jailbreaks non interceptés (FN global)** : prompts ayant réussi à passer à travers les deux filtres.

TABLE 1 – Performance du système de détection sur un échantillon de 218 prompts par modele de tes

	Détecteur seul	Système combiné
Taux de détection (TPR)	17.43%	**67.27%**
Faux négatifs	82.43%	**32.73%**

4.4 Résultats principaux

Comme indiqué dans le tableau 1, l'utilisation conjointe du détecteur et du juge permet une réduction significative des attaques non interceptées. Le juge permet notamment de compenser les failles du détecteur sur les attaques ambiguës ou obfusquées.

4.5 Comparaison entre modèles de test

FIGURE 2 – Comparaison de la vulnérabilité des modèles LLM testés (sur 200 prompts malveillants)

La figure 2 montre que certains modèles, comme *Phi-2*, sont plus facilement exploitables, tandis que d'autres, comme *LLaMA 3.2*, ont une résistance plus marquée face aux jailbreaks directs, mais peuvent échouer sur des formulations détournées. Ces résultats soulignent l'importance de tester les prompts sur plusieurs modèles pour établir une évaluation robuste.

4.6 Analyse qualitative des erreurs

Parmi les cas non détectés par le système combiné, nous avons observé :
— Des prompts très obfusqués utilisant des synonymes et une logique inversée.
— Des réponses techniquement neutres mais contenant des sous-entendus ou biais implicites.
— Des limitations du modèle juge sur des contenus générés ambigus.
Ces observations motivent l'introduction future de techniques de paraphrase inversée, de juges spécialisés, ou d'une mémoire contextuelle.

4.7 Performance et latence

Le temps moyen de traitement par prompt (détecteur + génération + jugement) est estimé à :
— $\sim$ 600 ms pour prompts jailbreak detectes par le detecteur.
— $\sim$ 600 ms pour prompts malveillants nécessitant traitement complet et pour les prompts benignes.

L'infrastructure basée sur `Ollama` en local permet de maintenir une latence basse, compatible avec une utilisation interactive.

4.8 Limites actuelles

Notre système présente quelques limites notables :
— Il ne capture pas encore les attaques multi-tours avec état conversationnel.
— Les modèles juge et détecteur peuvent parfois produire des verdicts divergents sur des cas ambigus.
— Tous les modèles utilisés sont en anglais ; une adaptation multilingue est à envisager.

5 Discussion et Perspectives

Les résultats expérimentaux montrent que l'architecture hybride proposée, combinant un détecteur de prompts malveillants et un juge de réponses, permet de réduire de manière significative le taux de jailbreaks non détectés. L'intégration de plusieurs modèles LLMs dans la phase de test nous a permis de comparer leur robustesse face à des tentatives de contournement. Parmi les modèles évalués, certains comme llama ont montré une meilleure résistance aux attaques, tandis que d'autres comme mistral ont présenté une plus grande permissivité dans certains scénarios, mettant en évidence la variabilité des politiques de sécurité internes selon les architectures.

Une observation importante concerne la complémentarité entre les deux niveaux de défense. Le détecteur, bien que performant, reste perfectible lorsqu'il est confronté à des techniques d'évasion élaborées. L'ajout du juge permet alors une couche de validation a posteriori des réponses, renforçant ainsi la robustesse globale du système. Cependant, cette approche induit une latence supplémentaire, qu'il conviendra d'optimiser dans des déploiements en temps réel.

Par ailleurs, les résultats suggèrent qu'un même prompt peut provoquer des comportements très différents selon le modèle interrogé. Cela soulève la nécessité d'adapter les systèmes de détection aux spécificités de chaque LLM, et d'éviter les approches trop généralistes.

Perspectives

Plusieurs axes de recherche peuvent être envisagés à la suite de cette étude :
— **Amélioration du détecteur par apprentissage continu :** intégrer un mécanisme de réentraînement automatique basé sur les décisions du juge pour renforcer sa précision au fil du temps.
— **Enrichissement du corpus d'attaque :** exploiter des ensembles de prompts adverses plus variés, incluant des techniques de jailbreak nouvelles ou plus subtiles.
— **Diversification des modèles de test et du juge :** inclure davantage de modèles de familles différentes (Mistral, Mixtral, Falcon, Claude) pour analyser leur résilience face aux mêmes attaques.
— **Déploiement dans des environnements réels :** tester l'architecture dans un environnement applicatif concret (chatbot public, assistant interne sécurisé) pour évaluer sa scalabilité et sa pertinence opérationnelle.
— **Détection de jailbreaks indirects :** explorer des méthodes pour identifier les cas où la réponse semble neutre mais conduit à des comportements malveillants indirects (via code, suggestion, ou redirection).

En définitive, notre approche ouvre la voie à des architectures défensives dynamiques et adaptatives contre les attaques de jailbreak, un enjeu majeur pour la fiabilité et l'éthique des systèmes basés sur les LLMs.

6 Travaux futurs

Les résultats obtenus à travers notre architecture hybride constituent une base prometteuse pour la sécurisation des interactions avec les modèles de langage. Toutefois, plusieurs pistes peuvent être explorées pour approfondir et étendre notre travail.

- **Entraînement sur des données multi-modèles :** à l'heure actuelle, notre détecteur est entraîné principalement sur des prompts génériques. Un enrichissement du jeu de données avec des exemples spécifiques à différents modèles (e.g., LLaMA, Mistral, Falcon, GPT-NeoX) pourrait permettre une meilleure généralisation et une sensibilité accrue aux spécificités des LLMs testés.
- **Détection de jailbreaks implicites :** au-delà des réponses manifestement malveillantes, certains modèles peuvent générer des contenus dangereux de manière indirecte, subtile ou ambiguë. Intégrer des modules sémantiques ou des graphes de connaissances pourrait permettre de mieux identifier ces cas ambigus.
- **Apprentissage actif avec rétroaction du juge :** une boucle d'apprentissage active pourrait être introduite dans le système, dans laquelle le modèle juge fournit des retours d'entraînement au détecteur. Cela favoriserait l'amélioration continue de la détection, en adaptant le classifieur aux nouveaux types de prompts évasifs.
- **Analyse temporelle des attaques :** étudier l'évolution des techniques de jailbreak dans le temps à partir des prompts collectés permettrait de mieux anticiper les tendances adverses et de renforcer la résilience du système face à des attaques émergentes.
- **Évaluation de la robustesse sous contrainte d'adversaire adaptatif :** il serait pertinent de simuler un environnement où l'attaquant ajuste ses stratégies en réponse au comportement du système (attaque adaptative), afin d'évaluer la robustesse réelle du détecteur et du juge.
- **Déploiement à grande échelle et tests en environnement réel :** une intégration dans des systèmes réels (chatbots, plateformes de support client, assistants IA) permettrait de mesurer l'impact de notre architecture sur l'expérience utilisateur, la latence, et la détection de cas concrets de non-conformité.

Ces pistes ouvrent la voie à une recherche plus approfondie sur la prévention des abus dans les LLMs, en combinant apprentissage machine, cybersécurité, et traitement du langage naturel. À terme, une standardisation de ce type d'architecture défensive pourrait contribuer à définir des normes de sécurité pour les applications basées sur les modèles de langage.

7 Conclusion

Dans cet article, nous avons présenté une architecture hybride visant à renforcer la sécurité des modèles de langage de grande taille (LLMs) face aux attaques de type jailbreak. Notre approche repose sur une double barrière : un modèle détecteur entraîné pour filtrer les prompts suspects, suivi d'un modèle juge chargé d'évaluer les réponses générées lorsqu'un prompt semble bénin. Cette combinaison permet de réduire significativement les risques de contournement des protections intégrées aux LLMs.

En intégrant plusieurs modèles de test, notre système offre une vision comparative de la robustesse des différents LLMs face à des attaques adverses. Les résultats expérimentaux montrent une amélioration notable de la détection des interactions malveillantes, tout en conservant une faible perturbation des requêtes légitimes. Le tableau de bord développé permet une visualisation en temps réel de l'activité du système, facilitant ainsi l'analyse des performances et des comportements suspects.

Nos contributions ouvrent la voie à des systèmes plus sûrs et dynamiques dans le domaine de l'intelligence artificielle générative. Cette architecture modulable peut être déployée dans divers contextes applicatifs, allant des assistants virtuels aux plateformes de traitement automatisé du

langage, et peut s'adapter à l'évolution rapide des stratégies d'attaque.

Des travaux futurs viendront enrichir ce dispositif, notamment par l'optimisation du détecteur, la prise en compte d'attaques implicites, et la mise en place d'un apprentissage adaptatif basé sur les retours du juge. Ces perspectives s'inscrivent dans une volonté de proposer des fondations solides pour un usage responsable, éthique et sécurisé des LLMs.