Actes de CORIA-TALN-RJCRI-RECITAL 2025
Marseille, France, 30 juin au 4 juillet 2025

Évaluation Comparative de la Génération Contrainte vs. du Post-Parsing pour l'Analyse de Contenu par LLM : Étude sur le Corpus EUvsDisinfo

Kevin Séjourné[1] Marine Foucher[2] Alexandru Lata[1] Jean-Fabrice Lebraty[2]
(1) Cloud Temple
[kevin.sejourne, alexandru.lata]@cloud-temple.com
(2) Laboratoire de Recherche Magellan, iaelyon
[marine.foucher, jean-fabrice.lebraty]@univ-lyon3.fr

RÉSUMÉ

Les Grands Modèles de Langage (LLM) sont de plus en plus intégrés dans des applications nécessitant des sorties formatées. Deux approches principales existent : instruire le LLM de générer directement la structure (e.g., JSON, SQL) puis la parser (post-parsing), ou utiliser des techniques de génération contrainte garantissant la syntaxe. Cette étude compare rigoureusement ces deux méthodes sur une tâche d'analyse de désinformation à grande échelle (17k documents du corpus EUvsDisinfo) en utilisant quatre LLM (Llama-3.3 70B, DeepSeek R1 70B, Qwen 72B, Gemma 3 27B) et plusieurs températures de génération. Nos résultats indiquent que la génération contrainte offre une fiabilité syntaxique quasi parfaite, tandis que le post-parsing est opérationnellement plus robuste mais génère davantage d'erreurs de formatage.

ABSTRACT

Structured Generation vs. Post-Parsing : A Comparative Study for Content Analysis by LLM on the EUvsDisinfo Corpus

Large Language Models (LLM) are increasingly integrated into applications requiring formatted outputs. Two main approaches exist : instructing the LLM to directly generate the structure (e.g., JSON, SQL) and then parse it (post-parsing), or using constrained generation techniques guaranteeing syntax. This study rigorously compares these two methods on a large-scale disinformation analysis task (17k documents from the EUvsDisinfo corpus) using four LLM (Llama-3.3 70B, Deepseek R1 70B, Qwen 72B, Gemma 3 27B) and several generation temperatures. Our results indicate that constrained generation offers near-perfect syntactic reliability, while post-parsing is operationally more robust but generates more formatting errors.

MOTS-CLÉS : Grands Modèles de Langage, Génération Structurée, Désinformation, Analyse de Contenu, Évaluation.

KEYWORDS: Large Language Models, Structured Generation, Disinformation, Content Analysis, Evaluation.

ARTICLE : **Soumis à** CORIA-TALN 2025.

Actes de l'atelier Évaluation des modèles génératifs (LLM) et challenge 2025 (EvalLLM), pages 127–137.

1 Introduction

L'intégration des Grands Modèles de Langage (LLM) dans les applications professionnelles nécessite qu'ils produisent des sorties au format structuré et déterminé, facilement interprétables par des systèmes en aval. Cette exigence pose un défi majeur car les LLM sont conçus pour générer du texte libre plutôt que des formats rigides comme JSON ou XML.

Deux approches principales s'opposent pour obtenir des sorties structurées fiables : demander au LLM de générer directement une structure, puis la parser (Post-Parsing, PP), ou utiliser des techniques de génération contrainte (GC) garantissant la syntaxe dès la génération. Le choix entre ces méthodes implique des compromis entre simplicité d'implémentation, robustesse et qualité des résultats. La méthode PP est conceptuellement simple mais peut souffrir d'erreurs de formatage, tandis que la GC promet une conformité syntaxique parfaite.

Notre étude vise à répondre à la question : **quelle méthode est la plus fiable et performante pour obtenir des sorties structurées de LLM, en prenant pour cas d'application l'analyse de contenu de désinformation ?**

Le choix du corpus EUvsDisinfo est particulièrement pertinent dans ce contexte. Créé par l'East StratCom Task Force de l'Union Européenne, ce corpus documente systématiquement les cas de désinformation pro-Kremlin depuis 2015. Son utilisation dans notre étude répond à un double objectif : évaluer les capacités techniques des LLM sur des textes réels et complexes, tout en contribuant à un enjeu stratégique européen majeur - la lutte contre la désinformation. La nature sensible de ces données renforce également l'importance d'utiliser des modèles exécutables localement, sans dépendance à des API tierces.

Nous présentons une étude comparative à grande échelle utilisant ce corpus (17 700 documents), quatre LLM récents et cinq températures de génération, pour quantifier les compromis entre ces deux stratégies et offrir des recommandations pratiques aux développeurs.

2 État de l'Art sur la Génération de Sorties Structurées par les LLM

La capacité des Grands Modèles de Langage (LLM) à produire des sorties conformes à des formats structurés spécifiques (JSON, XML, listes, GeoJSON, etc.) est devenue un enjeu majeur pour leur déploiement effectif dans les applications logicielles et les flux de travail automatisés. Les travaux récents couvrent diverses facettes de cette problématique, allant de la caractérisation des besoins utilisateurs aux défis techniques, en passant par les méthodes d'amélioration, l'évaluation rigoureuse et les implications sécuritaires.

Besoin et Motivations Utilisateurs L'intégration des LLM requiert fréquemment une adhésion stricte à des formats prédéfinis. Une étude auprès de professionnels de l'industrie (Liu *et al.*, 2024a) a mis en évidence des besoins à la fois de bas niveau (format structuré, longueur) et de haut niveau (directives sémantiques, style, absence d'hallucinations). Satisfaire ces contraintes est crucial pour simplifier l'intégration et améliorer l'expérience utilisateur finale.

Défis et Performance Actuelle Malgré leurs capacités linguistiques, les LLM affichent des performances variables pour la génération structurée stricte (Li *et al.*, 2024b; Liu *et al.*, 2024b). Le parangon SoEval (Liu *et al.*, 2024b) montre que même les meilleurs modèles comme GPT-4 ont une marge de progression significative, notamment sur des tâches complexes ou multilingues. La génération de données spatiales structurées (e.g., GeoJSON) pose également des défis spécifiques (Li *et al.*, 2024a).

Techniques d'Amélioration Plusieurs stratégies sont explorées pour fiabiliser la sortie structurée. L'**Ingénierie des Prompts Spécifique** comme G&O (Li *et al.*, 2024b) sépare génération et formatage. Le **Décodage Contraint** (Geng *et al.*, 2025) force la conformité syntaxique via des frameworks dédiés (Guidance, Outlines). D'autres approches incluent le **Fine-tuning** et le **RAG** (Li *et al.*, 2024a), des **techniques historiques** comme l'**utilisation de structures** pour l'édition de connaissances (StruEdit (Bi *et al.*, 2024)) ou la restructuration de données (Ko *et al.*, 2024).

Évaluation Dédiée Les travaux récents reconnaissent le besoin de parangons spécifiques, avec SoEval (Liu *et al.*, 2024b) pour divers formats, JSONSchemaBench (Geng *et al.*, 2025) pour le décodage contraint via schémas JSON, et d'autres propositions pour données spatiales (Li *et al.*, 2024a). Toutefois ces évaluations restent sensibles à la formulation des prompts (Chu *et al.*, 2024).

Implications de Sécurité Une vulnérabilité critique, l'attaque par décodage contraint (CDA) (Zhang *et al.*, 2025), démontre que les mécanismes de sortie structurée peuvent être exploités pour contourner les gardes-fous des LLM en manipulant les grammaires (plan de contrôle) plutôt que les prompts (plan de données), soulignant un angle mort sécuritaire majeur.

3 Étude Expérimentale Comparative : Génération Structurée vs. Post-Parsing pour l'Analyse de Désinformation

3.1 Motivation et Pertinence de l'Étude

L'état de l'art met en lumière l'importance croissante des sorties structurées et l'émergence de techniques sophistiquées comme la génération contrainte par grammaire. Cependant, une approche pragmatique, largement adoptée en pratique, consiste simplement à *demander* au LLM de formater sa réponse en JSON (ou autre structure) et à utiliser ensuite un parseur externe, souvent basé sur des expressions régulières (regex), pour extraire les données. Cette méthode est perçue comme potentiellement plus simple à implémenter initialement, évitant la complexité de la mise en place de grammaires formelles ou de frameworks de décodage spécifiques.

Néanmoins, cette simplicité apparente peut se faire au détriment de la robustesse. Les LLM, en particulier avec des températures de génération non nulles, peuvent produire des variations inattendues dans leur formatage, même lorsqu'ils sont entraînés à suivre un schéma précis. Ces variations peuvent facilement "casser" un parseur regex rigide, menant à des échecs d'extraction de données. Inversement, si la génération contrainte garantit la validité syntaxique, elle pourrait potentiellement influencer la qualité sémantique du contenu ou introduire une latence supplémentaire.

Il existe donc un manque clair de comparaison empirique à grande échelle entre ces deux stratégies fondamentales :

1. **Post-Parsing :** Demander le format + extraire via regex/parsing.
2. **Génération Contrainte :** Forcer le format via une grammaire pendant la génération.

Cette étude est donc **judicieuse et pertinente** car elle vise à quantifier rigoureusement les compromis entre ces deux approches en termes de fiabilité (syntaxique et sémantique), de robustesse face à la variabilité (température, modèles), et d'efficacité (performance). Les résultats fourniront des informations précieuses aux praticiens pour choisir la stratégie la plus adaptée à leurs besoins concrets.

3.2 Objectifs Spécifiques

Notre étude vise à répondre aux questions suivantes :

1. Quelle méthode (Post-Parsing vs. Génération Contrainte) offre la meilleure *fiabilité* pour produire des sorties JSON valides et conformes au schéma requis ?
2. Comment la *température* de génération influence-t-elle la fiabilité et la cohérence des sorties pour chaque méthode ?
3. Y a-t-il des différences significatives de *performance* (latence, débit) entre les deux approches ?
4. Les conclusions sont-elles *généralisables* à travers différents LLM ?

3.3 Cadre Expérimental

Jeu de Données : EUvsDisinfo Nous utilisons le corpus public **EUvsDisinfo** [1] décrit par Leite *et al.* (2024). Ce jeu de données, principalement en anglais, regroupe environ 17723 analyses (6.5 Mo) de cas de désinformation pro-Kremlin menées par l'East StratCom Task Force de l'UE. Il offre un large volume de textes réels et variés, idéal pour tester les capacités d'analyse des LLM.

Tâche d'Analyse Pour chaque document, la tâche consiste à extraire ou inférer quatre informations et à les retourner dans un format JSON spécifique avec deux scores numériques (`toxicite`, `agressivite`) et deux champs textuels (`emotion`, `intention`). Cette tâche mixte représente un cas d'usage réaliste pour la génération structurée, où le modèle doit produire à la fois des valeurs numériques contraintes et du texte libre.

Exemple Concret d'Analyse Pour illustrer la tâche, voici un exemple réel de document du corpus EUvsDisinfo et la sortie JSON produite par un modèle :

Extrait du corpus (daté du 7 octobre 2023) :
"It is now possible to become a member of the European Union if only you can curry favour in the field of Russophobia."

Cette tâche combine à la fois l'évaluation quantitative (scores) et qualitative (analyse textuelle), représentant un cas d'usage réaliste pour l'évaluation automatique des sorties JSON des LLM.

1. https://euvsdisinfo.eu/fr/

```
{
"toxicite": "7", // <score entier 0-10>
"agressivite": "6", // <score entier 0-10>
"emotion": "Resentment", // <texte libre>
"intention": "Criticism/Accusation" // <texte libre>
}
```

FIGURE 1 – Exemple de sortie JSON pour l'analyse d'un document réel

Modèles Évalués Quatre LLM ouverts, sous forme quantifiée, sont utilisés :

— `neuralmagic/Llama-3.3-70B-Instruct-quantized.w8a8`
— `neuralmagic/DeepSeek-R1-Distill-Llama-70B-FP8-dynamic`
— `Qwen/Qwen2.5-VL-72B-Instruct-AWQ`
— `abhishekchohan/gemma-3-27b-it-quantized-W4A16`

Ce panel permet d'étudier l'influence de l'architecture, de la taille du modèle et des techniques de quantification. Notre choix délibéré de n'utiliser que des modèles exécutables localement répond à des impératifs de souveraineté numérique européenne. En effet, le Foreign Intelligence Surveillance Act (FISA) américain permet aux autorités d'accéder aux données traitées par des entreprises américaines, même lorsque ces données concernent des citoyens non-américains. L'analyse de la désinformation pro-Kremlin, sujet sensible pour la sécurité européenne, ne peut donc pas être confiée à des API tierces potentiellement soumises à ces législations extraterritoriales. Cette approche garantit également une maîtrise complète de la chaîne de traitement et une indépendance technologique essentielle pour ce type d'analyse stratégique.

Méthodes Comparées Deux méthodes sont comparées, implémentées dans des scripts distincts :

1. **Méthode 1 : Post-Parsing (PP)**
   — **Script :** Script Python dédié à l'analyse par post-parsing.
   — **Prompt :** Le prompt (identique pour les deux méthodes) instruit le LLM d'effectuer l'analyse et de répondre *uniquement* au format JSON avec les clés spécifiées (cf. Listing 1). Aucun mécanisme de contrainte n'est appliqué côté serveur.
   — **Parsing :** La sortie brute du LLM est traitée par une fonction Python (`parse_llm_response`) qui tente d'isoler le bloc JSON en supprimant les marqueurs de code (` ```json `, ` ``` `) avant d'utiliser `json.loads`. En cas d'échec, une erreur de parsing est enregistrée.
2. **Méthode 2 : Génération Contrainte (GC)**
   — **Script :** Script Python dédié à l'analyse par génération contrainte.
   — **Contrainte :** Utilisation du paramètre `response_format` de l'API compatible OpenAI, avec `type: "json_schema"`. Le schéma JSON cible (cf. Listing 1), incluant les clés et les types attendus (`integer`, `string`), est directement intégré dans la requête API envoyée au serveur vLLM.
   — **Prompt :** Le même prompt que pour la méthode PP est utilisé, demandant explicitement le format JSON en plus de l'application de la contrainte par schéma.
   — **Algorithme de contrainte :** Le module XGrammar (Dong *et al.*, 2025) propose à l'utilisateur de rédiger une grammaire formelle (CFG - Context-Free Grammar). Le module crée alors un automate à pile (PDA - Pushdown Automaton) qui vient masquer les tokens qui ne respectent pas la grammaire, forçant ainsi le respect de la structure définie. Le filtrage

est implémenté en attribuant des poids de $-\infty$ aux tokens invalides, les excluant de facto après l'application de la fonction softmax.

Nous utilisons un serveur vLLM (Kwon *et al.*, 2023) ; le module xgrammar [2] réalise la méthode GC dans vLLM.

Procédure Expérimentale Les expériences sont menées avec cinq températures de génération distinctes (T = {0.0 [3], 0.2, 0.4, 0.6, 0.8}), avec 5 exécutions indépendantes pour chaque combinaison modèle/température. Chaque exécution traite l'intégralité du corpus EUvsDisinfo (environ 17700 documents) par lots, totalisant 100 configurations expérimentales (4 modèles * 5 températures * 5 répétitions) pour chaque méthode.

3.4 Métriques d'Évaluation Automatique des Sorties JSON

Pour évaluer les deux approches, nous mesurons plusieurs métriques pour chaque méthode, modèle, température et exécution. Nous commençons par évaluer la **fiabilité syntaxique** en calculant le taux de JSON valides (après parsing pour PP, directement pour GC), le taux de conformité au schéma attendu (présence des clés requises, respect des types de données) et le taux de respect des contraintes de plage numérique pour les scores. Puis nous analysons la **cohérence et la qualité sémantiques** des sorties. Cela inclut la mesure de la cohérence inter-exécutions (en examinant la variance des scores numériques et la similarité textuelle des champs libres pour les températures supérieures à zéro) et le calcul du taux d'erreurs sémantiques, telles que les refus de réponse ou les sorties hors-sujet. Enfin, nous mesurons la **performance et l'efficacité** en termes de latence de traitement par document, de débit (documents traités par seconde) et du nombre moyen de tokens générés par réponse.

Pour quantifier la cohérence inter-exécutions, nous utilisons la divergence de Jensen-Shannon au carré (JSD^2) entre les distributions de catégories. Notre étude montre que l'évaluation des LLM requiert, au-delà des métriques de performance, une analyse de leur robustesse face aux variations de température et de modèles.

3.5 Contribution Attendue

Les résultats de cette étude fourniront une base empirique solide pour comparer les approches de Post-Parsing et de Génération Contrainte. Nous attendons des informations quantitatives sur leurs forces et faiblesses respectives en termes de fiabilité et d'efficacité, aidant les développeurs à faire des choix éclairés pour l'intégration des LLM dans des tâches nécessitant des sorties structurées.

4 Résultats

Cette section présente les résultats comparatifs entre la méthode Post-Parsing (PP) et la méthode de Génération Contrainte (GC) selon les métriques définies précédemment.

2. https ://github.com/mlc-ai/xgrammar
3. La "température à zéro" désigne l'algorithme glouton (greedy) sélectionnant le token le plus probable. Le déterminisme n'est pas parfait en raison de tokens équiprobables, d'effets de quantification et d'arrondis numériques.

4.1 Fiabilité Syntaxique et Conformité au Schéma

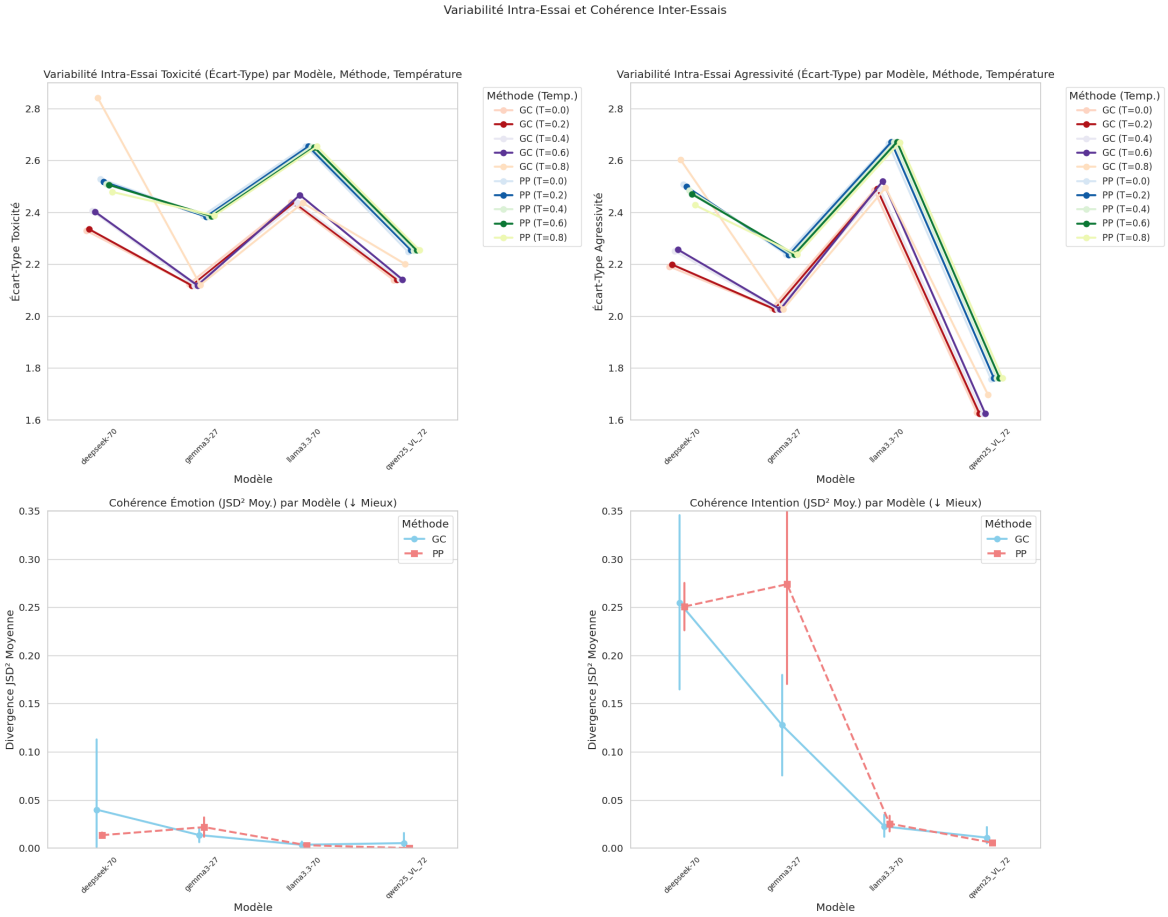

FIGURE 2 – Comparaison des distributions et de la consistance des résultats entre les méthodes PP et GC selon le modèle, la température pour 5 exécutions. L'analyse porte sur la dispersion des scores (toxicité, agressivité) et la variation des annotations textuelles (émotion, intention).

La première métrique évaluée est la capacité de chaque méthode à produire une sortie JSON syntaxiquement valide. Les quantités d'erreurs permettent d'apprécier les comparaisons suivantes sur le contenu.

Les données révèlent des différences majeures entre les deux approches : **Post-Parsing (PP)** présente un taux d'échec de parsing élevé (25% des documents) avec le modèle Qwen se distinguant par une robustesse supérieure, tandis que la **Génération Contrainte (GC)** élimine pratiquement les erreurs de formatage JSON (<1% même à T=0.8) *lorsque l'inférence aboutit*.

Les résultats des analyses montrent que GC a une meilleure cohérence inter-exécutions dans les annotations catégorielles (émotions et intentions), tandis que PP a une meilleure cohérence dans les évaluations numériques (toxicité et agressivité). De plus, GC a des taux d'erreur nettement inférieurs à ceux de PP. Enfin, GC montre un meilleur accord inter-modèles. La figure 2 illustre la distribution et la consistance des résultats pour chaque méthode, en fonction du modèle et de la température.

En résumé, GC assure une fiabilité syntaxique quasi parfaite, tandis que PP présente un taux d'erreur de formatage élevé quel que soit le modèle.

4.2 Qualité et Cohérence Sémantique

Au-delà de la validité syntaxique, nous évaluons la cohérence sémantique des informations extraites.

4.2.1 Cohérence Inter-Exécutions (Stabilité)

La stabilité des résultats à travers les exécutions indépendantes révèle des comportements distincts, comme le montre le tableau 1.

Méthode	Similarité texte	Variance Tox.	Variance Aggr.	Stabilité T↑
Post-Parsing (PP)	98%	0,018-0,019	0,024-0,025	Excellente
Génération Contrainte (GC)	83-98%	0,02-0,67	0,02-0,57	Variable

TABLE 1 – Comparaison de la cohérence inter-exécutions des méthodes PP et GC

Post-Parsing (PP) démontre une *stabilité de cohérence* exceptionnelle (98% de similarité) à travers toutes les températures, avec une variance des scores numériques extrêmement faible, tandis que la **Génération Contrainte (GC)** présente une cohérence plus variable selon la température - excellente à T=0.0 et T=0.6, mais se dégradant notablement à T=0.4 et T=0.8 (similarité réduite à 83%).

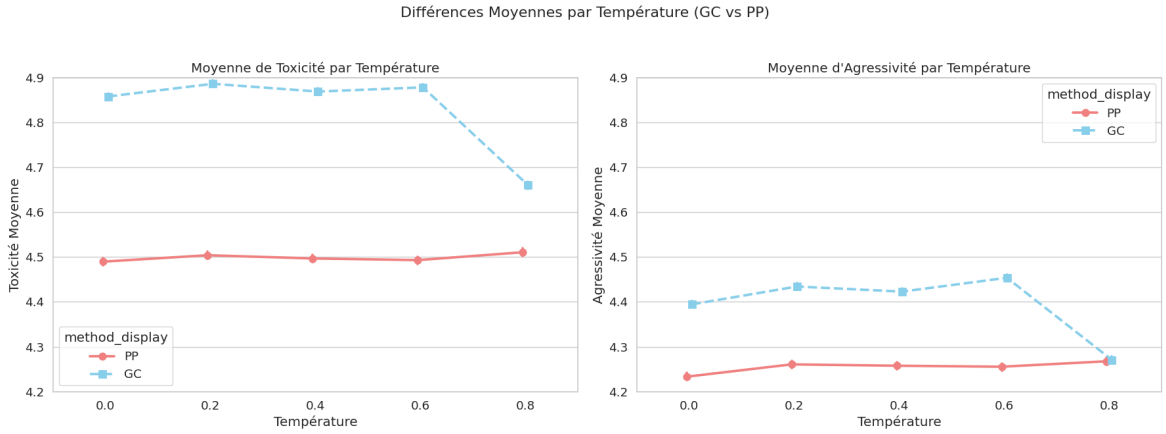

FIGURE 3 – Impact de la température sur les performances des deux méthodes

4.2.2 Taux d'Erreurs Sémantiques et Accord Inter-Modèles

La méthode PP présente un taux d'erreur sémantique stable (4,5-5%) pour les champs textuels, tandis que GC affiche un taux beaucoup plus faible (0-0,82%) sur les exécutions réussies. L'accord inter-modèles reste limité pour les deux méthodes, particulièrement pour les champs textuels (<25%).

La Fig. 3 illustre l'impact de la température sur les performances des deux méthodes. GC excelle pour son faible taux d'erreurs sémantiques, tandis que PP offre une cohérence plus stable à travers les températures.

4.3 Performance

Les performances d'inférence varient principalement selon la taille des modèles : Gemma (27B) atteint 3000 tokens/seconde, Llama 2000 T/s, Qwen 1500 T/s, et DeepSeek seulement 600 T/s pour des lots de 64. Aucune différence significative n'a été observée entre PP et GC, suggérant que l'application de contraintes par schéma JSON n'introduit pas de surcoût de calcul notable. De même, le sur-coût du *Parsing* PP est négligeable (µs).

5 Discussion et Analyse Comparative

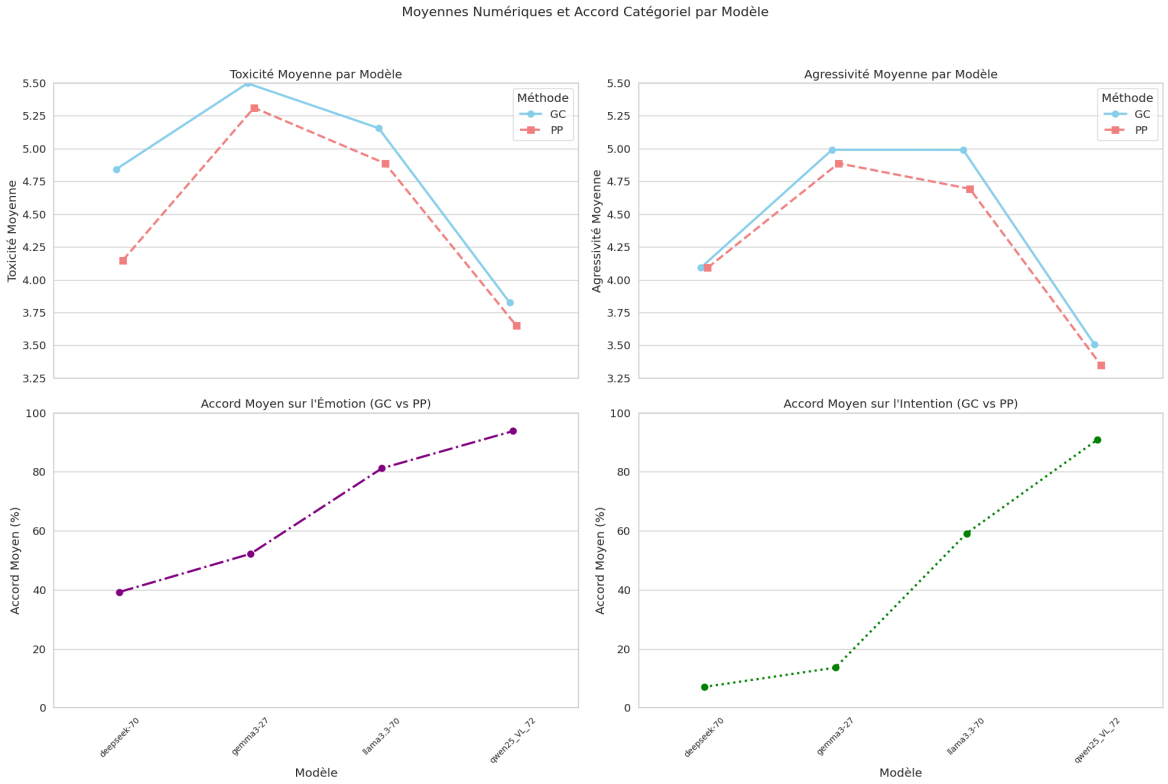

FIGURE 4 – Performances comparatives par modèle : Moyennes de toxicité et d'agressivité (GC vs PP) et Accord moyen sur l'émotion et l'intention (GC vs PP).

Les résultats expérimentaux offrent un éclairage nuancé sur le compromis entre PP et GC. La figure 4 illustre les différences de performance moyenne par modèle, montrant les moyennes de toxicité et d'agressivité pour chaque méthode, ainsi que l'accord moyen obtenu entre les méthodes pour l'émotion et l'intention. Ces résultats, combinés aux analyses de fiabilité et de cohérence, fournissent des enseignements pour l'évaluation des LLM en production.

Fiabilité syntaxique : GC produit des sorties JSON quasi parfaites[4] (<1% d'erreurs) quand elle fonctionne, surpassant PP (25% d'erreurs). La robustesse supérieure de Qwen avec PP suggère des différences notables entre les architectures de modèles.

4. Les rares échecs de GC (<1%) sont dus à des plantages du module de grammaire (xgrammar), des résultats vides, ou à une limite de caractères atteinte avant la fin de la structure JSON.

Cohérence et qualité sémantiques : Contre-intuitivement, PP produit des sorties plus stables à travers les températures que GC. La variance des scores et la similarité textuelle de PP restent constantes, démontrant une robustesse remarquable face aux variations de paramètres de génération. La GC se dégrade à certaines températures, mais conserve un avantage net sur le taux d'erreurs sémantiques (0-1% vs 5%).

Implications pratiques pour l'évaluation des LLM : Sur la base des résultats, la méthode GC semble être préférable en raison de sa meilleure cohérence dans les annotations catégorielles, de ses taux d'erreur inférieurs et de son meilleur accord inter-modèles. Cependant, la méthode PP peut être préférable si la cohérence des évaluations numériques est plus importante.

Nos expériences montrent que l'évaluation des LLM en production doit considérer non seulement les taux d'erreurs bruts, mais aussi la robustesse opérationnelle et la stabilité des résultats face aux variations de paramètres. Les travaux futurs devraient analyser les causes des échecs de GC, comparer d'autres frameworks, et évaluer des schémas plus complexes.

6 Conclusion et Perspectives d'Évaluation

Cette étude comparative a évalué deux approches pour obtenir des sorties JSON structurées à partir de LLM : le Post-Parsing (PP) et la Génération Contrainte (GC). Nos mesures quantitatives mettent en lumière un compromis entre fiabilité syntaxique, cohérence sémantique et robustesse opérationnelle - trois dimensions essentielles pour l'évaluation des LLM en production.

La GC démontre une fiabilité syntaxique quasi parfaite (>99%) pour les inférences réussies. Elle offre également un taux d'erreurs sémantiques remarquablement bas (0-0,82%) sur les exécutions réussies, témoignant d'une qualité de contenu supérieure.

PP se distingue par sa stabilité de cohérence exceptionnelle (98%) à travers toutes les températures testées, démontrant une résistance aux variations de paramètres que GC ne possède pas. Cependant, il génère un taux élevé d'erreurs de formatage (25%) et d'erreurs sémantiques (5%), nécessitant des mécanismes de correction.

Nos expériences fournissent des enseignements précieux sur l'évaluation des LLM en production, montrant que la méthode GC semble être préférable en raison de sa meilleure cohérence dans les annotations catégorielles, de ses taux d'erreur inférieurs et de son meilleur accord inter-modèles. Cependant, la méthode PP peut être préférable si la cohérence des évaluations numériques est plus importante.

Nos travaux futurs approfondiront l'évaluation automatique des sorties JSON en explorant les causes des échecs de GC, en comparant d'autres frameworks de génération contrainte, et en développant des stratégies de réparation pour les erreurs de formatage en PP.

Références

BI B., LIU S., WANG Y., MEI L., GAO H., FANG J. & CHENG X. (2024). Struedit : Structured outputs enable the fast and accurate knowledge editing for large language models. *arXiv preprint arXiv :2409.10132*.

CHU K., CHEN Y.-P. & NAKAYAMA H. (2024). A better llm evaluator for text generation : The impact of prompt output sequencing and optimization. *arXiv preprint arXiv :2406.09972*.

DONG Y., RUAN C. F., CAI Y., LAI R., XU Z., ZHAO Y. & CHEN T. (2025). Xgrammar : Flexible and efficient structured generation engine for large language models.

GENG S., COOPER H., MOSKAL M., JENKINS S., BERMAN J., RANCHIN N., WEST R., HORVITZ E. & NORI H. (2025). Generating structured outputs from language models : Benchmark and studies. *arXiv preprint arXiv :2501.10868*.

KO H., YANG H., HAN S., KIM S., LIM S. & HORMAZABAL R. (2024). Filling in the gaps : Llm-based structured data generation from semi-structured scientific data. In *ICML 2024 AI for Science Workshop*.

KWON W., LI Z., ZHUANG S., SHENG Y., ZHENG L., YU C. H., GONZALEZ J. E., ZHANG H. & STOICA I. (2023). Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

LEITE J. A., RAZUVAYEVSKAYA O., BONTCHEVA K. & SCARTON C. (2024). Euvsdisinfo : A dataset for multilingual detection of pro-kremlin disinformation in news articles. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, p. 5380–5384.

LI D., ZHAO Y., WANG Z., JUNG C. & ZHANG Z. (2024a). Large language model-driven structured output : A comprehensive benchmark and spatial data generation framework. *ISPRS International Journal of Geo-Information*, **13**(11), 405.

LI Y., RAMPRASAD R. & ZHANG C. (2024b). A simple but effective approach to improve structured language model output for information extraction. *arXiv preprint arXiv :2402.13364*.

LIU M. X., LIU F., FIANNACA A. J., KOO T., DIXON L., TERRY M. & CAI C. J. (2024a). " we need structured output" : Towards user-centered constraints on large language model output. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, p. 1–9.

LIU Y., LI D., WANG K., XIONG Z., SHI F., WANG J., LI B. & HANG B. (2024b). Are llms good at structured outputs ? a benchmark for evaluating structured output capabilities in llms. *Information Processing & Management*, **61**(5), 103809.

ZHANG S., ZHAO J., XU R., FENG X. & CUI H. (2025). Output constraints as attack surface : Exploiting structured generation to bypass llm safety mechanisms. *arXiv preprint arXiv :2503.24191*.