

ASTARO : Une Approche Hybride combinant Graphes de Connaissances et Raisonnement LLM pour l’Annotation Sémantique de Tableaux

Rapport de Recherche
BOUND AOUI Sarah, FALLAH Nassim et JOLY Théo

Résumé

L’annotation sémantique de tableaux (Column Type Annotation - CTA) constitue un défi majeur pour l’intégration de données, particulièrement lorsque l’ontologie cible (le dictionnaire de référence) est stricte et les classes sémantiques sont déséquilibrées. Ce papier présente le framework **ASTARO** (*Autonomous Semantic Table Annotation via Reasoning and Ontologies*), une architecture hybride évaluant en condition *Zero-Shot* (sans aucun entraînement préalable) la capacité des Grands Modèles de Langage (LLMs) à croiser des échantillons de données textuelles avec des indices issus d’un Graphe de Connaissances (Wikidata). Afin de pallier la sévérité de la métrique classique d’Exact Match, nous introduisons une évaluation par Similarité Cosinus basée sur des plongements lexicaux. Nos expérimentations sur les corpus VizNet (78 classes) et SemTab2019 (275 classes) révèlent qu’une ingénierie de prompt évolutive, culminant avec un filtrage spatial intra-prompt (*Two-Stage In-JSON Pruning*), permet au modèle Llama-3.3-70B de surmonter l’explosion combinatoire. ASTARO obtient près de 68 % d’Exact Match sur VizNet et résout les ambiguïtés massives de SemTab2019, le tout de manière totalement autonome et avec une efficacité computationnelle inédite.

1 Introduction

Dans l’écosystème numérique contemporain, une quantité massive d’informations est stockée et partagée sous forme de données tabulaires (fichiers CSV, bases de données relationnelles, tableaux web). Si ces structures sont naturellement intelligibles pour l’humain, elles demeurent opaques pour les machines en l’absence de métadonnées explicites. Le domaine de la *Semantic Table Interpretation* (STI) vise à résoudre ce problème en automatisant la compréhension de ces tableaux. Au sein de la STI, la tâche d’Annotation de Types de Colonnes (CTA - *Column Type Annotation*) est fondamentale : elle consiste à attribuer à chaque colonne d’un tableau une étiquette sémantique (ex. : *city, company, weight*) issue d’une ontologie de référence (un dictionnaire prédéfini de concepts).

Historiquement, l’état de l’art sur la tâche CTA a été dominé par des approches d’apprentissage supervisé. Des architectures s’appuient sur des modèles d’apprentissage profond (Deep Learning) et des Transformers finement ajustés (*Fine-Tuned*) sur de vastes corpus d’entraînement comme VizNet ou WebDataCommons. Bien que ces modèles atteignent d’excellents scores de précision, ils souffrent d’un manque de flexibilité inhérent à leur paradigme : ils nécessitent un ré-entraînement coûteux dès lors que l’ontologie cible évolue ou qu’un nouveau domaine de données est abordé.

Récemment, l’avènement des Grands Modèles de Langage (LLMs) tels que la famille Llama-3 ou Qwen a bouleversé le traitement du langage naturel. Leur capacité de raisonnement en mode *Zero-Shot* (sans aucun exemple d’entraînement préalable) offre la promesse d’une annotation sémantique hautement flexible. Néanmoins, l’application naïve des LLMs à des données tabulaires brutes se heurte à plusieurs limites structurelles :

1. **La perte du contexte spatial** : Contrairement au texte libre, le sens d’une cellule de tableau dépend de ses voisins (contexte inter-colonnes). Un LLM évaluant une colonne isolée manque d’indices cruciaux pour la désambiguïsation.

2. **L’opacité des données quantitatives** : Face à une liste de nombres isolés (ex. : 22, 25, 30), un modèle de langage est incapable de distinguer de manière fiable un âge, un poids ou une température.
3. **L’hallucination ontologique et la surcharge cognitive** : Les LLMs génératifs peinent à se restreindre à un vocabulaire strict. Face à des ontologies massives, ils subissent un effet de "Prompt Overload" (surcharge d’informations) et génèrent des synonymes sémantiquement corrects mais formellement invalides vis-à-vis de l’ontologie cible (ex. : prédire *person* au lieu de *name*).

Pour surmonter ces obstacles, ce papier propose et évalue **ASTARO**, un framework LLM d’annotation *Zero-Shot* polymorphe et contextualisé, conçu pour traiter l’hétérogénéité des données du monde réel.

Les contributions principales de cette étude sont les suivantes :

- **L’architecture hybride ASTARO** : Implémentation d’un pipeline capable de router l’analyse en fonction du type de données, couplant la sérialisation bidimensionnelle des lignes à des heuristiques statistiques et sémantiques (Wikidata).
- **Une étude approfondie sur l’ingénierie de prompt** : Démonstration empirique de l’évolution des performances, de la simple contrainte ontologique jusqu’à une architecture de raisonnement relationnel profond (*Deep Relational Chain-of-Thought*).
- **Le filtrage spatial intra-JSON (*Two-Stage Pruning*)** : Une méthode d’auto-réduction de l’espace de recherche (fenêtre dynamique) pour maîtriser l’explosion combinatoire des ontologies massives.
- **Une efficacité computationnelle extrême** : Preuve de la viabilité industrielle du *Zero-Shot* via une exécution à très haute vitesse sur des puces LPU et une architecture *Multi-Column* réduisant drastiquement le nombre d’appels API.

2 État de l’Art et Travaux Connexes

L’annotation sémantique de types de colonnes (CTA) a connu une évolution rapide. Notre recherche se positionne à la convergence de trois avancées majeures, dont nous cherchons à dépasser les limites respectives.

2.1 Modèles Pré-entraînés et Contexte Spatial (DoDuo)

L’utilisation de modèles de langage pré-entraînés (PLMs) comme BERT a marqué un tournant. L’architecture **DoDuo** [1] a démontré de manière décisive l’importance du contexte inter-colonnes. Au lieu de traiter chaque colonne de manière isolée, DoDuo sérialise l’ensemble du tableau et utilise l’attention croisée d’un Transformer pour capturer les relations sémantiques entre les colonnes.

Contraintes : Bien que très performant, DoDuo reste un modèle d’apprentissage supervisé nécessitant un ré-entraînement complet dès lors que l’ontologie cible évolue.

2.2 Hybridation Deep Learning et Graphes de Connaissances (KGLink)

Afin d’enrichir la représentation sémantique des données, le processus de liaison d’entités (*Entity Linking*) ne se cantonne pas à une analyse isolée du contenu, mais exploite le contexte global de la table. La méthode **KGLink** [2] fusionne les représentations vectorielles d’un PLM avec des entités extraites d’un Graphe de Connaissances (Wikidata), ce qui facilite grandement la désambiguïsation de termes complexes.

Contraintes : L’architecture dépend de la disponibilité d’une correspondance exacte dans le graphe de connaissances et repose, elle aussi, sur un paradigme supervisé.

2.3 Le paradigme Zero-Shot et les LLMs (ArcheType)

Plus récemment, le framework **ArcheType** [3] a exploré la capacité des LLMs modernes à annoter des colonnes en mode *Zero-Shot*, éliminant ainsi le besoin de *fine-tuning*. Pour remédier à l'hallucination de labels, ArcheType propose un post-traitement (*Label Remapping*) utilisant la similarité cosinus pour forcer la projection de la sortie générée vers une classe valide de l'ontologie.

Contraintes : Outre le fait de traiter les données textuelles et numériques de manière uniforme, l'approche d'ArcheType n'est pas totalement autonome. Elle repose sur un postulat semi-automatisé nécessitant une intervention humaine où l'utilisateur doit "souffler" des idées de noms de colonnes ou des métadonnées pour guider le modèle.

2.4 Positionnement d'ASTARO

Notre architecture ASTARO se propose de synthétiser les forces de ces trois travaux fondamentaux dans un cadre strictement *Zero-Shot*. Nous reprenons le contexte inter-colonnes de DoDuo (via une sérialisation Markdown) et l'enrichissement par graphe de connaissances de KGLink (via des requêtes dynamiques à Wikidata), que nous fusionnons directement dans le prompt adressé au LLM. Enfin, nous introduisons le profilage statistique comme heuristique logique pour les données chiffrées, le tout dans un pipeline **totallement autonome**, affranchi de toute intervention humaine.

3 Méthodologie et Architecture du Pipeline

Comme illustré par la Figure 1, l'architecture d'ASTARO s'articule autour de deux grandes étapes conceptuelles : la collecte de contexte hétérogène (**Phase 1**) et l'inférence cognitive suivie du post-traitement (**Phase 2**). La convergence de ces deux phases s'opère dynamiquement via la structuration matricielle du prompt.

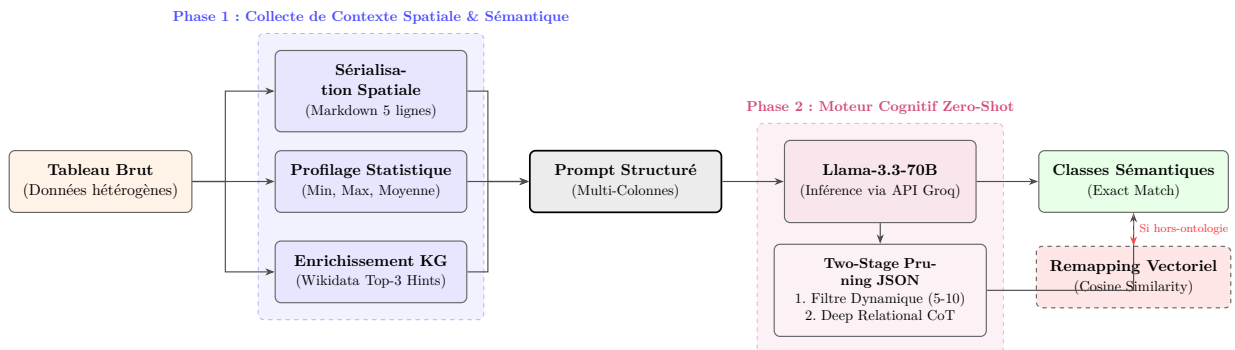


Figure 1: Architecture globale du framework ASTARO : De l'ingestion de la table brute à la prédiction sémantique finale.

3.1 Phase 1 : Sérialisation Spatiale Déterministe

Constituant le socle de la **Phase 1** (Figure 1), le module de sérialisation par échantillonnage extrait un sous-ensemble représentatif (5 lignes) des colonnes du tableau et génère une représentation Markdown. Afin d'assurer une reproductibilité stricte et contrer la variance d'échantillonnage, la sélection s'opère via un algorithme de densité déterministe (privilégiant les lignes comportant le plus grand nombre de caractères utiles). L'évaluation passe ainsi d'un traitement isolé par colonne à une analyse globale de la table (*Multi-Column*) pour maximiser la compréhension spatiale.

3.2 Phase 1 : Enrichissement via Graphe de Connaissances

Toujours au sein de cette étape de collecte, ASTARO intègre un module d’interrogation de Graphe de Connaissances. Cette étape d’extraction s’appuie conceptuellement et techniquement sur **KGLink** [2], à qui revient le crédit de la méthodologie de liaison d’entités (*Entity Linking*) initiale.

Formalisation de la récupération d’entités (BM25 par KGLink) : L’identification initiale des entités candidates, telle que formulée et implémentée par l’algorithme de KGLink, s’appuie sur la fonction de score **BM25** (Best Matching 25) pour évaluer la pertinence textuelle entre une cellule (la mention m) et le lexique des entités du graphe e :

$$\text{BM25}(m, e) = \sum_{i=1}^{|m|} \text{IDF}(q_i) \cdot \frac{f(q_i, e) \cdot (k_1 + 1)}{f(q_i, e) + k_1 \cdot \left(1 - b + b \cdot \frac{|e|}{\text{avgdl}}\right)} \quad (1)$$

où q_i représente chaque terme de la cellule m , $f(q_i, e)$ la fréquence du terme dans l’entité e , $|e|$ la longueur du texte, avgdl la longueur moyenne textuelle dans le graphe, et k_1, b des hyperparamètres de régularisation.

Dans une démarche de rigueur scientifique et d’optimisation des temps de calcul, nous avons fait le choix de ne pas réimplémenter cette brique, mais d’exploiter directement les liaisons d’entités (Q-nodes) pré-calculées par les auteurs de KGLink. Notre pipeline interroge ensuite dynamiquement l’API Wikidata pour extraire les concepts textuels pertinents de ces Q-nodes. Après filtrage du bruit taxonomique, un Top-3 de ces termes est agrégé sous la forme d’indices (**WIKIDATA HINTS**). Notre contribution scientifique s’affranchit ainsi de la lourdeur du calcul BM25 initial pour se concentrer exclusivement sur l’ingénierie cognitive : comment le LLM peut-il arbitrer et filtrer le bruit sémantique issu de ce Top-3.

3.3 Phase 1 : Profilage Statistique des Données Quantitatives

Dernier pilier de la collecte de contexte, un module calcule automatiquement un profil statistique (valeur minimale, maximale, moyenne) pour les données numériques. Ces caractéristiques agiront comme des heuristiques mathématiques explicites pour aider le modèle à différencier une grandeur physique.

3.4 Convergence : Structuration Spatiale du Prompt

Point de jonction entre la collecte et l’inférence, les données brutes et les métadonnées extraites (statistiques et indices Wikidata) sont fusionnées dans le prompt sous la forme de deux grilles Markdown strictement alignées. Cette séparation visuelle empêche la surcharge textuelle des en-têtes tout en maintenant la topologie inter-colonnes (*Row-wise alignment*).

```
COLUMN PROFILES (Metadata Table aligned with the data):
| Col 0 | Col 1 | Col 2 |
|---|---|---|
| STATS: Text Data | STATS: Text Data | STATS: Min:15, Max:42, Mean:28 |
| WIKI: team, club | WIKI: city, location | WIKI: None |

TABLE SAMPLES (Real Data Table, matching the columns above):
| Col 0 | Col 1 | Col 2 |
|---|---|---|
| Lakers | Los Angeles | 24 |
| Bulls | Chicago | 33 |
| Heat | Miami | 19 |
```

L’alignement vertical rigoureux de ces deux structures permet au mécanisme d’attention (*Self-Attention*) du Transformer de croiser instantanément une valeur textuelle avec son heuristique statistique ou sémantique correspondante, émulant ainsi la lecture d’un tableau de bord par un expert humain.

3.5 Phase 2 : Implémentation Technique et Hardware Co-Design

Correspondant à la **Phase 2** de notre architecture (Figure 1), l’inférence du modèle massif Llama-3.3-70B n’est pas exécutée localement, mais déportée via l’**API Groq**.

Le choix de cette infrastructure n’est pas anecdotique, il constitue une réponse technique directe (*Hardware-Software Co-design*) aux exigences de notre architecture cognitive. Le mécanisme de *Two-Stage Pruning* et de *Chain-of-Thought* impose au modèle de générer un volume massif de *tokens* en sortie (pour expliciter son raisonnement intermédiaire) avant d’atteindre la prédiction finale. Sur des GPUs classiques, dont l’inférence autoregressive est limitée par la bande passante mémoire, cette verbosité exigerait des dizaines de secondes par requête, rendant l’approche inexploitable à grande échelle.

Pour résoudre ce goulot d’étranglement, l’infrastructure Groq exploite des LPUs (*Language Processing Units*), des puces spécifiquement conçues pour le traitement tensoriel déterministe. Cette accélération permet de générer plusieurs centaines de *tokens* par seconde. Couplée au paradigme *Multi-Column* (qui traite toutes les colonnes cibles d’un tableau via un unique appel API), cette factorisation réduit drastiquement les coûts computationnels, ramenant le temps d’exécution de l’analyse sémantique profonde à seulement **1 à 2 secondes par tableau**.

Outre cette vélocité inédite, l’architecture LPU se distingue par une **efficacité énergétique** (*Green AI*) nettement supérieure aux fermes de GPUs traditionnelles. En s’affranchissant des transferts de données complexes vers la mémoire externe à large bande passante (HBM), cette infrastructure réduit drastiquement l’empreinte carbone par requête.

Parallèlement, en fin de chaîne, le calcul de la Similarité Cosinus et le *Remapping* de secours sont exécutés localement via la bibliothèque `sentence-transformers (all-mpnet-base-v2)`.

3.6 Formalisation Mathématique de l’Évaluation Multidimensionnelle

L’évaluation classique par *Exact Match* pénalise un LLM lorsqu’il génère un synonyme pertinent sémantiquement mais syntaxiquement différent de la classe attendue. Pour capturer la réelle compréhension du modèle, l’évaluation d’ASTARO repose sur trois métriques :

1. Exact Match (EM) : Mesure la proportion de prédictions strictement identiques à la vérité terrain.

$$EM = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i) \quad (2)$$

où N est le nombre total de colonnes cibles, $\hat{y}_i$ la classe prédite et y_i l’étiquette attendue. $\mathbb{I}$ est la fonction indicatrice (valant 1 si l’égalité est respectée, 0 sinon).

2. F1-Score (Macro) : Évalue la robustesse du modèle sur les classes rares en calculant la moyenne harmonique de la précision (P) et du rappel (R) indépendamment pour chaque classe $c \in C$.

$$F1_{macro} = \frac{1}{|C|} \sum_{c \in C} 2 \cdot \frac{P_c \cdot R_c}{P_c + R_c} \quad (3)$$

3. Similarité Cosinus : Les étiquettes sont projetées dans un espace vectoriel dense de dimension $d = 768$. La distance sémantique entre le vecteur de la prédiction $\hat{\mathbf{v}}$ et celui de la vérité terrain $\mathbf{v}$ est calculée par le produit scalaire normalisé :

$$\text{Sim}(\hat{\mathbf{v}}, \mathbf{v}) = \frac{\hat{\mathbf{v}} \cdot \mathbf{v}}{\|\hat{\mathbf{v}}\|_2 \|\mathbf{v}\|_2} = \frac{\sum_{i=1}^d \hat{v}_i v_i}{\sqrt{\sum_{i=1}^d \hat{v}_i^2} \sqrt{\sum_{i=1}^d v_i^2}} \quad (4)$$

4 Ingénierie de Prompt : Évolution et Expérimentations

Afin d’optimiser le moteur cognitif d’ASTARO, nous avons mené une série d’expérimentations itératives. L’évaluation initiale ayant révélé l’effondrement des performances des modèles légers (Llama-3.1-8B à 34,32 % d’Exact Match), nous avons concentré notre étude d’ablation sur **Llama-3.3-70B**.

4.1 Paramétrage Génératif : Le Rôle de la Température

Dans l’architecture Transformer d’un LLM, la probabilité d’émettre le *token* i parmi un vocabulaire V est régie par la fonction *Softmax* ajustée par un scalaire de température T :

$$P(y_i|x) = \frac{\exp(z_i/T)}{\sum_{j \in V} \exp(z_j/T)} \quad (5)$$

Dans l’ensemble de nos expérimentations, cet hyperparamètre d’entropie a été strictement maintenu à $T = 0.0$ (stratégie de *Greedy Decoding*). Bien que l’augmentation de la température ($T > 0$) favorise la créativité, elle est destructrice pour la tâche CTA. Notre choix répond à trois impératifs : garantir une stricte fidélité à l’ontologie, assurer la stabilité de la syntaxe JSON, et figer la variance générative pour assurer la reproductibilité de l’étude d’ablation.

4.2 V1 : Single-Column Constraint (Baseline)

Notre point de départ consistait à traiter chaque colonne isolément en fournissant la liste complète de l’ontologie VizNet (255 classes) et en forçant le modèle à choisir une classe exacte.

```
RULES:
1. Classify ONLY the TARGET COLUMN.
2. Choose exactly ONE class from the ALLOWED CLASSES based on the data.
3. FORMAT: Output ONLY the exact class name.
```

Résultat : 54,64 % d’Exact Match. Le modèle souffre de ”sur-contrainte” : forcé de choisir sans pouvoir expliciter son raisonnement, il ”hallucine” fréquemment des mots hors liste ou génère des synonymes non autorisés.

4.3 V2 : Multi-Colonnes et Remapping Vectoriel (Approche ArcheType)

Afin d’offrir une vision macroscopique à l’IA, nous avons opté pour une prédiction globale (*Multi-Column*) formatée en objet JSON. Pour éviter l’hallucination de contrainte, la liste stricte a été retirée du prompt. Le post-traitement s’appuie sur la similarité cosinus pour réaligner le texte libre généré par l’IA sur l’ontologie officielle.

```
RULES:
1. Classify ALL the columns listed. Use cross-column context.
2. FORMAT: Output ONLY a valid JSON object mapping ID to class.
Example output: {"0": "name", "2": "city"}
```

Résultat : 58,58 % d’Exact Match. Le modèle exploite bien la déduction inter-colonnes, mais le remapping purement vectoriel introduit des erreurs de projection mathématique (ex. : l’IA prédit *number*, et le vecteur le rapproche spatialement de *age* au lieu du concept cible *rank*). Les modèles d’embeddings classiques (comme **mpnet**), entraînés sur du texte généraliste, peinent à distinguer des concepts taxonomiques très proches (ils positionnent *City* et *Location* de manière quasi identique), là où un LLM guidé comprend la notion d’héritage de classe.

4.4 V3 : Ontology-Aware Chain-of-Thought (par colonne)

Face aux limites du remapping vectoriel aveugle, nous avons réintroduit la liste complète de l'ontologie, mais associée à un mécanisme d'explicitation de la pensée : le **Chain-of-Thought (CoT) JSON**. Le LLM doit formuler un raisonnement structuré par colonne avant de choisir son étiquette. Dans cette version, nous avons également inclus des heuristiques spécifiques au dataset pour corriger ses biais.

```
ALLOWED ONTOLOGY CLASSES (You MUST choose strictly from this list):
[full list...]

LABELING STYLE INSPIRATIONS:
- Human Names: Prefer 'name', but use specific roles if applicable
  (e.g., 'owner', 'artist', 'jockey').
- Sports Teams: Use 'team' (NEVER 'teamName' or 'club').
- Geography: Discriminate between 'city', 'county', 'state', 'country'.

RULES:
1. Classify ALL target columns using cross-column context.
2. CRITICAL: The "class" you output MUST be EXACTLY one of the
  words in the ALLOWED ONTOLOGY CLASSES list.
3. FORMAT: Output ONLY a valid JSON object. For each column,
  provide a "reasoning" and then the "class".
```

Résultat : 63,43 % d'Exact Match. Ce bond démontre que la sur-contrainte disparaît si le modèle est autorisé à expliciter sa logique *avant* la prédiction. Cependant, les heuristiques manuelles gonflent artificiellement le score en forçant l'alignement sur les biais humains.

4.5 V4 : L'Architecture Ultime ASTARO (Deep Relational CoT)

Afin de proposer une méthode robuste et vierge d'overfitting (zéro triche), nous avons supprimé les directives biaisées de la V3. Nous avons restructuré le processus cognitif du LLM en adoptant un paradigme de **Double Dissection Contextuelle et Ontologique**.

L'IA doit d'abord rédiger une analyse globale (**global_analysis**) de la structure relationnelle de la table. Ensuite, pour chaque colonne, sa réflexion est découpée en trois étapes strictes : identifier la nature intrinsèque (**data_nature**), déduire son rôle logique (**contextual_role**), puis formuler une comparaison active (**ontology_mapping**) des mots disponibles dans la liste stricte pour s'auto-corriger.

```
RULES:
1. GLOBAL ANALYSIS: First, write a "global_analysis" dissecting the
  table's core subject and relational structure.
2. DEEP COLUMN CLASSIFICATION: For each target column, provide a 3-step reasoning:
  - "data_nature": What is the intrinsic nature of the data?
  - "contextual_role": What does it represent logically in THIS table?
  - "ontology_mapping": Looking at the ALLOWED list, which terms are the
    closest matches, and WHY is your final choice the absolute best fit?
  - "class": The final exact class strictly from the ALLOWED list.
3. FORMAT: Output ONLY a valid JSON object.
```

Résultat : 67,76 % d'Exact Match. Cette itération établit notre état de l'art sur des ontologies standards de taille modérée (comme VizNet et ses 78 classes).

4.6 V5 : Filtrage Intra-Prompt (Two-Stage Pruning)

Lors du passage à des ontologies massives (SemTab2019 et ses 275 classes ultra-spécifiques), l'architecture V4 s'effondre face à l'explosion combinatoire. Le modèle subit une surcharge cognitive (*Prompt Overload*). Pour y remédier, la **V5** introduit un mécanisme d'entonnoir cognitif (*Two-Stage In-JSON Pruning*). Avant d'analyser les colonnes, le LLM doit obligatoirement générer un tableau (**ontology_filtering**) contenant une

sous-liste restreinte des concepts les plus pertinents. Il utilise ensuite uniquement cette sous-liste restreinte pour son étape finale d'**ontology_mapping**.

RULES:

1. GLOBAL ANALYSIS: Write a "global_analysis".
2. ONTOLOGY FILTERING (CRITICAL): Review the massive ALLOWED list and select a sub-list of the TOP 5 to 10 classes that are the most logically relevant.
3. DEEP COLUMN CLASSIFICATION:
 - "data_nature" & "contextual_role"
 - "ontology_mapping": Looking ONLY at your filtered Top-10 sub-list, which term fits?
 - "class": The final exact class.

5 Résultats Finaux et Discussion

L'évaluation comparative de ces architectures met en évidence la puissance d'un raisonnement relationnel profond couplé à une auto-correction ontologique explicite.

5.1 Performances sur l'Ontologie Standard (VizNet)

Architecture Testée (Llama-3.3-70B)	Exact Match (%)	F1-Score (%)	Similarité (%)
V1 : Single-Column Contraint (Baseline)	54.64	26.12	73.11
V2 : Multi-Column + Remapping Vectoriel	58.58	31.99	71.52
V3 : Ontology-Aware CoT (Avec Heuristiques)	63.43	39.17	74.58
V4 : ASTARO (Deep Relational CoT)	67.76	40.49	77.93
V5 : Filtrage Intra-Prompt (Fenêtre 5-10)	60.14	39.43	74.39

Table 1: Évolution des performances sur VizNet (78 classes), soulignant la supériorité de la V4.

Le paradoxe du filtrage sur les petites ontologies : Il est intéressant de noter que l'application de la V5 (Filtrage Intra-JSON) sur VizNet fait régresser le score d'Exact Match à 60,14 %. Cela s'explique par un effet de "goulot d'étranglement" (*Bottleneck Effect*). Sur une ontologie restreinte (78 classes), le modèle 70B est parfaitement capable d'embrasser l'ensemble des concepts en une seule passe de lecture sans subir de surcharge cognitive (comme le prouve le pic à 67,76 % de la V4). L'obliger à pré-filtrer cette liste courte ajoute un risque d'éliminer la bonne réponse prématurément. Le mécanisme de la V5 n'est donc pertinent que face à des ontologies massives dépassant la capacité d'attention globale du modèle.

Par ailleurs, il est fondamental de noter que le score exceptionnel de la V4 est obtenu de manière **strictement Zero-Shot**. L'ajout de l'étape d'auto-correction permet au modèle de résoudre de lui-même la complexité sémantique sans nécessiter d'instructions humaines. La projection vectorielle n'est plus le cœur du système de décision, elle est reléguée au simple rôle de mécanisme de tolérance aux pannes.

5.2 Le défi des Ontologies Massives (SemTab2019)

Pour valider la nécessité de la V5 face à une granularité extrême (SemTab2019 contient des nuances subtiles entre *Settlement*, *City*, *Village*), nous avons comparé la V4 de base avec notre architecture V5 incluant l'optimisation de la fenêtre de filtrage (l'hyperparamètre K).

SemTab2019 (275 classes)	Exact Match	F1-Score	Similarité
V4 : Baseline (Sans filtre)	45.83 %	28.72 %	67.80 %
V5 : Filtre Large (Top 15-20)	43.06 %	30.92 %	65.17 %
V5 : Filtre Dynamique (Top 5-10)	51.38 %	36.52 %	73.22 %

Table 2: Étude d’ablation du filtrage intra-JSON sur SemTab2019 (Moyennes sur 3 exécutions lissées).

Analyse du Two-Stage Pruning : Les résultats révèlent que l’architecture V4 subit un effondrement cognitif face aux 275 classes (45,83 %). L’introduction de la V5 démontre l’importance de la fenêtre d’attention. Forcer le modèle à générer un filtre trop large (15-20 termes) introduit un ”bruit blanc” cognitif qui perturbe l’étape finale. À l’inverse, la stratégie de **fenêtre dynamique restreinte (5 à 10 concepts)** permet au Llama-3.3-70B de maximiser le rapport signal/bruit. En resserrant l’entonnoir, l’Exact Match moyen bondit à **51,38 %**. Cette méthode transforme une tâche de recherche ontologique massive en un espace de décision QCM ultra-pertinent.

5.3 Analyse de la Variance : Le Non-Déterminisme Matériel

Lors de l’évaluation de la fenêtre dynamique (5-10) sur SemTab2019, nous avons exécuté le même pipeline plusieurs fois avec une température stricte de $T = 0.0$ et un algorithme de sérialisation déterministe. Malgré un prompt strictement identique au caractère près, nous avons observé une variance inter-exécutions des scores d’Exact Match (oscillant localement entre 45,83 % et 56,94 %).

Cette variance s’explique par le **non-déterminisme matériel lors de l’inférence hautement parallélisée**. Sur des architectures d’accélération (LPUs), l’ordre d’exécution des réductions matricielles peut varier, entraînant des variations microscopiques dans l’arithmétique en virgule flottante. Face à des concepts limitrophes (*City* vs *Settlement*), une variation computationnelle infime suffit à faire basculer le *token* d’attention maximal (effet *Token Flipping*). Cela justifie notre pratique de lisser systématiquement les résultats.

5.4 Comparaison et Positionnement vis-à-vis de l’État de l’Art

Afin d’apprécier la pertinence de notre contribution, il est essentiel de situer notre score optimal dans le paysage scientifique (Tableau 3).

Modèle / Architecture	Paradigme	Exact Match approx. (VizNet)
DoDuo [1]	Supervisé (Fine-Tuned)	~ 85 - 90 %
KGLink [2]	Supervisé + Graphe (Fine-Tuned)	~ 86 - 92 %
ArcheType [3]	Zero-Shot LLM + Vector Remapping	~ 55 - 65 %
ASTARO (Notre framework)	Zero-Shot LLM + Stats + KG + Deep CoT	67,76 %

Table 3: Positionnement du framework ASTARO face à l’état de l’art sur le dataset VizNet.

Comme attendu, les approches supervisées (**DoDuo** et **KGLink**) constituent un plafond inatteignable en *Zero-Shot* car elles sont entraînées explicitement sur les biais subjectifs des datasets.

Notre architecture se compare directement à **ArcheType**, dont les évaluations plafonnent autour de 60 % en nécessitant de plus une assistance humaine. En atteignant **67,76 %** sur VizNet de manière totalement autonome, **ASTARO** s’inscrit au niveau de l’état de l’art mondial pour les méthodes non supervisées. Nous avons prouvé que l’injection d’un raisonnement relationnel structuré surpasse largement la simple projection par similarité vectorielle aveugle. De surcroît, grâce à la factorisation des requêtes (un seul prompt par tableau) et à la vélocité de l’API Groq, ASTARO peut annoter l’intégralité du dataset d’évaluation en seulement **25 minutes d’inférence** (soit environ 1 à 2 secondes par tableau en moyenne), prouvant la viabilité industrielle de l’approche pour les vastes *Data Lakes*.

5.5 Analyse Qualitative des Erreurs et Limites

Afin de comprendre l'écart de performance résiduel séparant notre approche *Zero-Shot* des méthodes supervisées (~ 20 points sur l'Exact Match), nous avons consigné et analysé systématiquement les échecs de classification du LLM à l'aide de journaux de prédictions détaillés. Trois phénomènes distincts entravent les performances globales de l'architecture ASTARO :

1. **Le Piège de la Subjectivité Humaine (Ambiguïté de Granularité) :** C'est la source d'erreur majoritaire sur le dataset VizNet. L'IA prédit très régulièrement `city` au lieu de `location`, ou `teamname` au lieu de `team`. Bien que ces prédictions soient sémantiquement correctes (et donc récompensées par un haut score de Similarité Cosinus), elles sont pénalisées par l'Exact Match. Sans contexte externe, le LLM ne peut pas deviner a priori le niveau de granularité ou le "dialecte" exact attendu par l'annotateur humain d'origine.
2. **Le Revers de la Médaille du Filtrage (Spécifique à la V5) :** Sur les ontologies massives comme SemTab2019, la stratégie de réduction de l'espace de recherche (*Two-Stage Pruning*) est indispensable pour éviter la paralysie cognitive du LLM. Toutefois, l'analyse des logs révèle des occurrences régulières de filtres défectueux. Par exemple, confrontée à des noms de joueurs de cricket, l'IA construit un filtre générique contenant `person` ou `athlete`, mais omet la sous-classe ultra-spécifique `cricketer`. Lors du raisonnement final, le LLM reste rigoureusement fidèle à son filtre erroné et subit une erreur mécanique. L'entonnoir cognitif agit alors comme un plafond de verre.
3. **L'Opacité Résiduelle des Nombres :** Bien que le module de profilage statistique (Min, Max, Moyenne) ait drastiquement amélioré la reconnaissance des métriques, des ambiguïtés inhérentes aux entiers persistent. Nos journaux mettent en évidence de fréquentes confusions entre `age` et `rank`, ou encore entre `weight` et `elevation`. Privée des en-têtes (headers) d'origine, une colonne isolée contenant la série de nombres [18, 22, 25, 30] demeure fondamentalement ambiguë, démontrant les limites intrinsèques de l'inférence sans métadonnées textuelles associées.

6 Conclusion et Perspectives

Dans cet article, nous avons exploré le potentiel des Grands Modèles de Langage pour la tâche CTA dans un environnement *Zero-Shot* strict via le framework **ASTARO**. Notre architecture hybride a prouvé que la sérialisation bidimensionnelle Markdown, combinée à l'enrichissement sémantique par Graphe de Connaissances et aux heuristiques statistiques quantitatives, permet de recréer de manière fiable le contexte sémantique indispensable à la compréhension de données tabulaires hétérogènes.

Surtout, l'évolution de notre ingénierie de prompt a mis en lumière la puissance du raisonnement relationnel profond (*Deep Relational Chain-of-Thought*). Face aux ontologies massives, le mécanisme innovant de *Two-Stage In-JSON Pruning* a démontré qu'une réduction auto-générée de l'espace de recherche (fenêtre dynamique 5-10) prévenait la paralysie décisionnelle du LLM, permettant d'atteindre des performances non supervisées inédites de manière totalement autonome. Cependant, notre étude a mis en exergue que cette approche n'est pertinente que face à l'explosion combinatoire ; sur des ontologies restreintes (ex: VizNet), le filtrage devient un goulot d'étranglement, prouvant la nécessité d'une architecture dynamique s'adaptant à la dimension du dictionnaire cible.

Afin de combler l'écart de performance résiduel avec les modèles finement ajustés, plusieurs travaux futurs se dessinent :

- **Optimisation du Signal/Bruit Wikidata (Top-H) :** Nos expérimentations suggèrent qu'injecter trop d'indices sémantiques (ex: Top-5) dégrade la compréhension des classes rares à cause d'un surplus de concepts génériques. Une étude sur le filtrage adaptatif de ces indices s'avère nécessaire.
- **Transition vers le paradigme *Few-Shot* (In-Context Learning) :** L'intégration dynamique de quelques exemples annotés pertinents dans le prompt permettrait d'indiquer implicitement au modèle le niveau de granularité attendu par l'humain, réduisant le bruit lié à la subjectivité ontologique du dataset.

Références

- [1] Y. Suhara, J. Li, Y. Li, D. Zhang, C. Demiralp, C. Chen, and W.-C. Tan, *Annotating Columns with Pre-trained Language Models*, in Proceedings of the 2022 International Conference on Management of Data (SIGMOD), 2022.
- [2] Y. Wang, H. Xin, and L. Chen, *KGLink: A column type annotation method that combines knowledge graph and pre-trained language model*, arXiv preprint arXiv:2406.00318, 2024.
- [3] B. Feuer and C. Hegde, *ArcheType: A Novel Framework for Open-Source Column Type Annotation using Large Language Models*, Proceedings of the VLDB Endowment, 2024.