Orange Research
Shape the future and value innovation

Sur la génération de grands graphes de connaissances synthétiques à l'aide de modèles de langage (LLM) via une approche Ontology2Graph

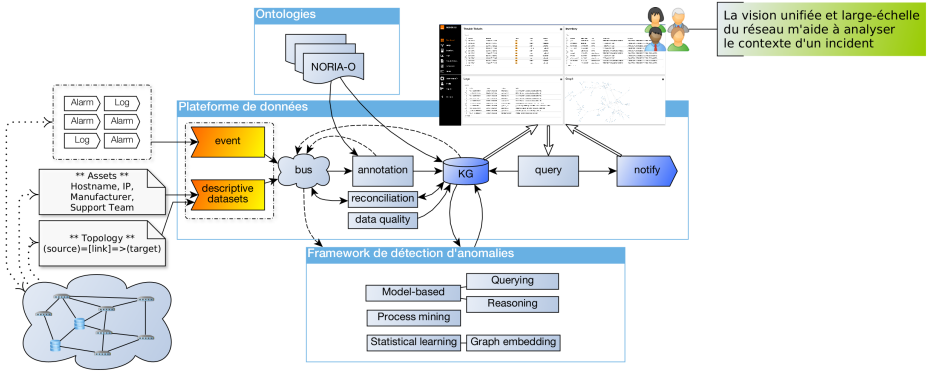
IC – PFIA 2026

Philippe Dooze, Lionel Tailhardat, Ovidiu Pascal

Orange

2 juillet 2026

Contexte – Disposer simplement de Graphes de Connaissances (GdC)



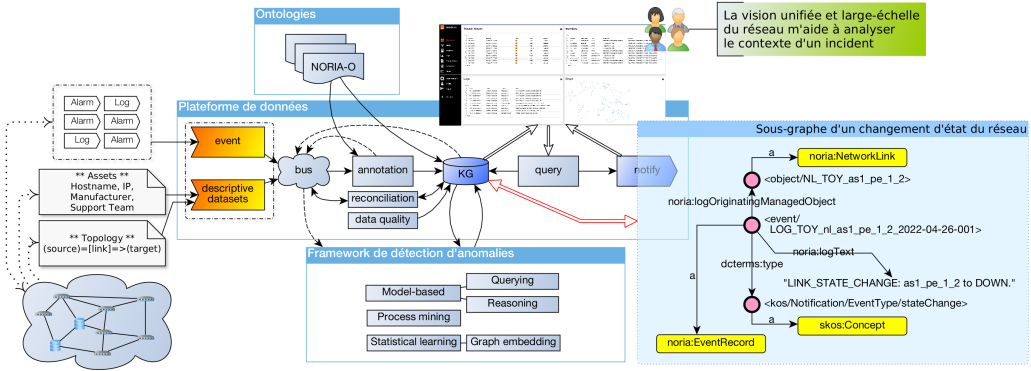
Applications Médecine (James'24), bibliométrie (Francart'14), **réseaux et cybersécurité** (Tailhardat'24, Nikolentzos'23), etc.

Caractéristiques Ordre et taille importante ; structure logique (ontologie). CS-KG 2.0 : 25M nœuds, 67M arrêtes (Dessi'25).

Cycle de vie 1) prototypage

→ Génération de Graphes de Connaissances Synthétiques (GCS). Graphe structuré par un vocabulaire formel, avec données fictives.

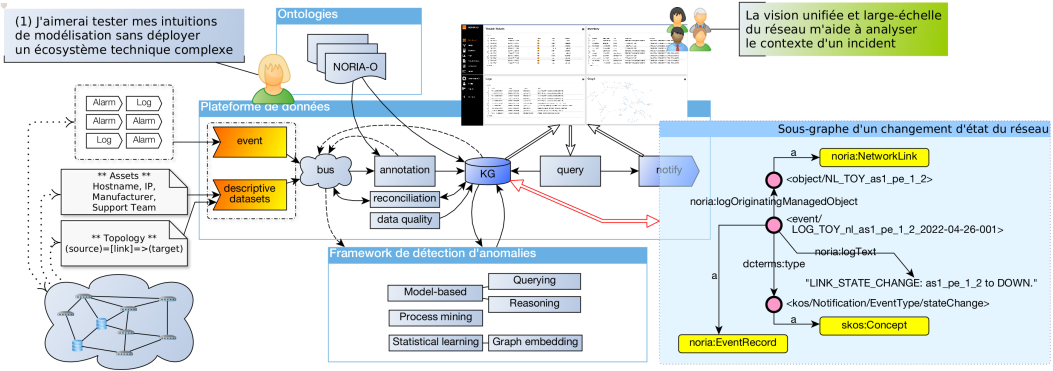
Contexte – Disposer simplement de Graphes de Connaissances (GdC)



Applications Médecine (James'24), bibliométrie (Francart'14), **réseaux et cybersécurité** (Tailhardat'24, Nikolentzos'23), etc.

Caractéristiques Ordre et taille importante ; structure logique (ontologie). CS-KG 2.0 : 25M nœuds, 67M arrêtes (Dessi'25).

Contexte – Disposer simplement de Graphes de Connaissances (GdC)



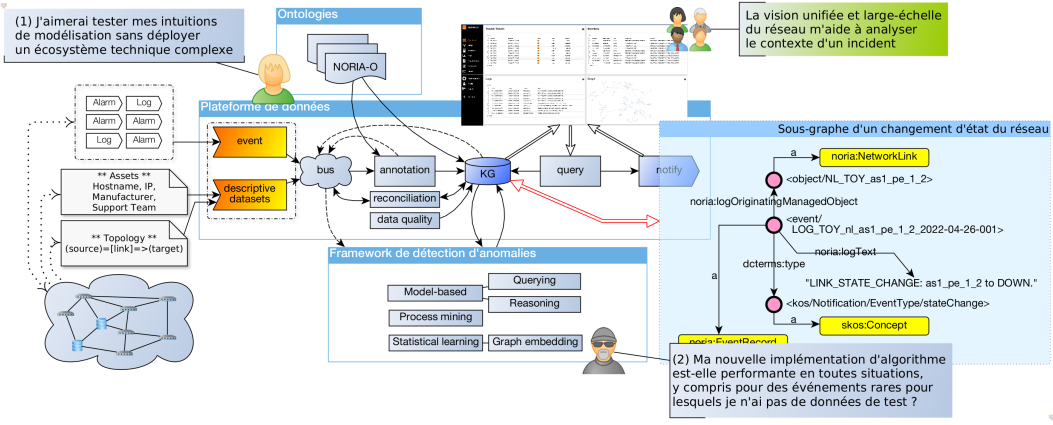
Applications Médecine (James'24), bibliométrie (Francart'14), **réseaux et cybersécurité** (Tailhardat'24, Nikolentzos'23), etc.

Caractéristiques Ordre et taille importante ; structure logique (ontologie). CS-KG 2.0 : 25M nœuds, 67M arrêtes (Dessi'25).

Cycle de vie 1) prototypage 2) test des pipelines d'IA, 3) partage.

→ Génération de Graphes de Connaissances Synthétiques (GCS). Graphe structuré par un vocabulaire formel, avec données fictives.

Contexte – Disposer simplement de Graphes de Connaissances (GdC)

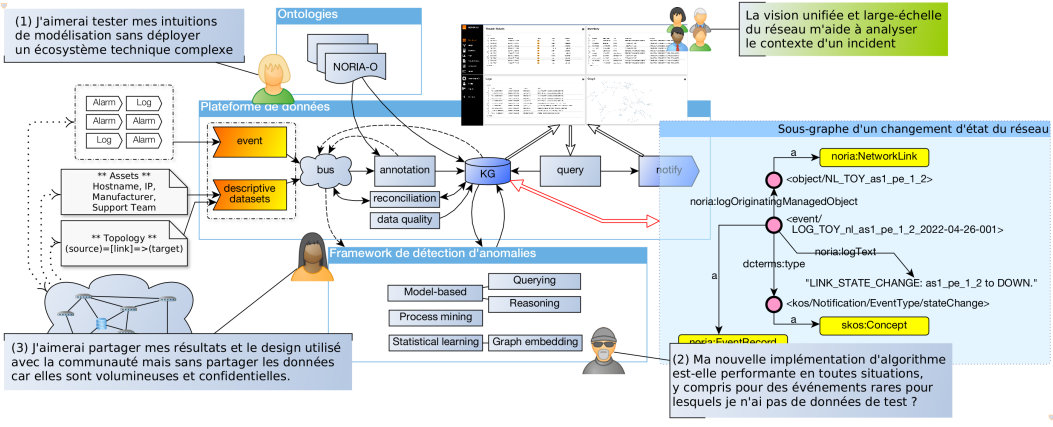


Applications Médecine (James'24), bibliométrie (Francart'14), **réseaux et cybersécurité** (Tailhardat'24, Nikolentzos'23), etc.

Caractéristiques Ordre et taille importante ; structure logique (ontologie). CS-KG 2.0 : 25M nœuds, 67M arrêtes (Dessi'25).

Cycle de vie 1) prototypage, 2) test des pipelines d'IA

Contexte – Disposer simplement de Graphes de Connaissances (GdC)



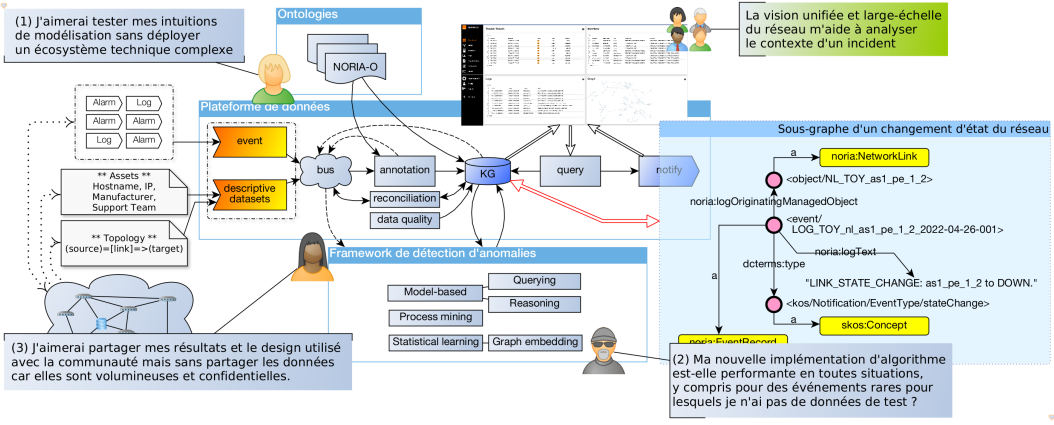
Applications Médecine (James'24), bibliométrie (Francart'14), **réseaux et cybersécurité** (Tailhardat'24, Nikolentzos'23), etc.

Caractéristiques Ordre et taille importante ; structure logique (ontologie). CS-KG 2.0 : 25M nœuds, 67M arrêtes (Dessi'25).

Cycle de vie 1) prototypage, 2) test des pipelines d'IA, 3) partage.

→ Génération de Graphes de Connaissances Synthétiques (GCS). Graphe structuré par un vocabulaire formel, avec données fictives.

Contexte – Disposer simplement de Graphes de Connaissances (GdC)



Applications Médecine (James'24), bibliométrie (Francart'14), **réseaux et cybersécurité** (Tailhardat'24, Nikolentzos'23), etc.

Caractéristiques Ordre et taille importante ; structure logique (ontologie). CS-KG 2.0 : 25M nœuds, 67M arrêtes (Dessi'25).

Cycle de vie 1) prototypage, 2) test des pipelines d'IA, 3) partage.

↪ Génération de Graphes de Connaissances Synthétiques (GCS). Graphe structuré par un vocabulaire formel, avec données fictives.

Etat de l'art – Méthodes et types de graphes pour produire des GCS

Solution	Méthode	GSF	GDC
NetworkX	Stochastique	✓	✗
igraph	et déterministe	✓	✗
Pygraft (Pascal'26)	Stochastique	✗	✓
FG-difusion (Bufort'25)	Modèle d' IA	✗	✓
(Nikolentzos'23)	spécifique	✗	✓
GraphGpt	Text2Graph	✓	✗
(Fu'25)	et LLM existant	✗	✓

GSF Graphe Sans Formalisme spécifique

- Modèles paramétriques stochastiques (Erdős-Rényi, Barabási-Albert).
- GSF → GDC : transformation de données (table de correspondance avec une ontologie).

GDC Graphe De Connaissances

- Stochastique : sans garanties sur la présence de structures spécifiques au domaine.
- Réseaux de neurones : données d'apprentissage + infrastructure complexe et coûteuse.
- LLM : extension de graphe existant ou traduction "Text2Graph" sans rapport à une ontologie de référence.

Complexité Forte adhérence entre l'approche choisie et le domaine de données + utilisation de pipelines sophistiqués.

Ouverture Les LLMs récents sont capables d'interpréter une description générale d'un problème et y répondre dans un format spécifique grâce au prompt engineering.

↪ Approfondir l'usage des LLMs pour produire des GCS ?

Etat de l'art – Méthodes et types de graphes pour produire des GCS

Solution	Méthode	GSF	GDC
NetworkX	Stochastique	✓	✗
igraph	et déterministe	✓	✗
Pygraft (Pascal'26)	Stochastique	✗	✓
FG-difusion (Bufort'25)	Modèle d' IA	✗	✓
(Nikolentzos'23)	spécifique	✗	✓
GraphGpt	Text2Graph	✓	✗
(Fu'25)	et LLM existant	✗	✓

GSF Graphe Sans Formalisme spécifique

- Modèles paramétriques stochastiques (Erdős-Rényi, Barabási-Albert).
- GSF → GDC : transformation de données (table de correspondance avec une ontologie).

GDC Graphe De Connaissances

- Stochastique : sans garanties sur la présence de structures spécifiques au domaine.
- Réseaux de neurones : données d'apprentissage + infrastructure complexe et coûteuse.
- LLM : extension de graphe existant ou traduction "Text2Graph" sans rapport à une ontologie de référence.

Complexité Forte adhérence entre l'approche choisie et le domaine de données + utilisation de pipelines sophistiqués.

Ouverture Les LLMs récents sont capables d'interpréter une description générale d'un problème et y répondre dans un format spécifique grâce au prompt engineering.

↪ Approfondir l'usage des LLMs pour produire des GCS ?

Etat de l'art – Méthodes et types de graphes pour produire des GCS

Solution	Méthode	GSF	GDC
NetworkX	Stochastique	✓	✗
igraph	et déterministe	✓	✗
Pygraft (Pascal'26)	Stochastique	✗	✓
FG-difusion (Bufort'25)	Modèle d' IA	✗	✓
(Nikolentzos'23)	spécifique	✗	✓
GraphGpt	Text2Graph	✓	✗
(Fu'25)	et LLM existant	✗	✓

GSF Graphe Sans Formalisme spécifique

- Modèles paramétriques stochastiques (Erdős-Rényi, Barabási-Albert).
- GSF → GDC : transformation de données (table de correspondance avec une ontologie).

GDC Graphe De Connaissances

- Stochastique : sans garanties sur la présence de structures spécifiques au domaine.
- Réseaux de neurones : données d'apprentissage + infrastructure complexe et coûteuse.
- LLM : extension de graphe existant ou traduction "Text2Graph" sans rapport à une ontologie de référence.

Complexité Forte adhérence entre l'approche choisie et le domaine de données + utilisation de pipelines sophistiqués.

Ouverture Les LLMs récents sont capables d'interpréter une description générale d'un problème et y répondre dans un format spécifique grâce au prompt engineering.

↪ Approfondir l'usage des LLMs pour produire des GCS ?

Etat de l'art – Méthodes et types de graphes pour produire des GCS

Solution	Méthode	GSF	GDC
NetworkX	Stochastique	✓	✗
igraph	et déterministe	✓	✗
Pygraft (Pascal'26)	Stochastique	✗	✓
FG-difusion (Bufort'25)	Modèle d' IA	✗	✓
(Nikolentzos'23)	spécifique	✗	✓
GraphGpt	Text2Graph	✓	✗
(Fu'25)	et LLM existant	✗	✓

GSF Graphe Sans Formalisme spécifique

- Modèles paramétriques stochastiques (Erdős-Rényi, Barabási-Albert).
- GSF → GDC : transformation de données (table de correspondance avec une ontologie).

GDC Graphe De Connaissances

- Stochastique : sans garanties sur la présence de structures spécifiques au domaine.
- Réseaux de neurones : données d'apprentissage + infrastructure complexe et coûteuse.
- LLM : extension de graphe existant ou traduction "Text2Graph" sans rapport à une ontologie de référence.

Complexité Forte adhérence entre l'approche choisie et le domaine de données + utilisation de pipelines sophistiqués.

Ouverture Les LLMs récents sont capables d'interpréter une description générale d'un problème et y répondre dans un format spécifique grâce au prompt engineering.

↪ Approfondir l'usage des LLMs pour produire des GCS ?

Problématique – Forces et faiblesses des LLMs pour produire des GCS

LLMs et GCS Peu de travaux scientifiques abordent directement le sujet des GCS.

Text2Graph (Wang'25) variante "OIE-based" (Open Information Extraction) s'appuie uniquement sur le texte fourni en entrée pour extraire les entités et leurs relations ; "ontology-driven" fait intervenir un texte contraint par une ontologie associée.

Intuition Produire un GCS, c'est s'aligner sur les caractéristiques essentielles d'un domaine de discours, et donc sur une ontologie → solliciter un LLM pour générer un GCS à partir d'une ontologie et de consignes fournies en entrée.

Ontology2Graph – Evaluer la capacité des LLMs à produire des GCS

- **QR. 1** Capacité des LLMs à générer de grands graphes (en ordre et taille)
- **QR. 2** Cohérence de ces graphes avec l'ontologie en question et le domaine associé, notamment en ce qui concerne l'instanciation de classes d'objets sémantiquement valides.

Hypothèse L'ontologie fournit suffisamment d'informations pour que le LLM, grâce à ses données d'entraînement et à un prompt adapté, produise un GCS correctement instancié (càd. répondant à des critères de syntaxe, de cohérence sémantique et d'utilisabilité).

Problématique – Forces et faiblesses des LLMs pour produire des GCS

LLMs et GCS Peu de travaux scientifiques abordent directement le sujet des GCS.

Text2Graph (Wang'25) variante "OIE-based" (Open Information Extraction) s'appuie uniquement sur le texte fourni en entrée pour extraire les entités et leurs relations ; "ontology-driven" fait intervenir un texte contraint par une ontologie associée.

Intuition Produire un GCS, c'est s'aligner sur les caractéristiques essentielles d'un domaine de discours, et donc sur une ontologie → solliciter un LLM pour générer un GCS à partir d'une ontologie et de consignes fournies en entrée.

Ontology2Graph – Evaluer la capacité des LLMs à produire des GCS

QR. 1 Capacité des LLMs à générer de grands graphes (en ordre et taille)

QR. 2 Cohérence de ces graphes avec l'ontologie en question et le domaine associé, notamment en ce qui concerne l'instanciation de classes d'objets sémantiquement valides.

Hypothèse L'ontologie fournit suffisamment d'informations pour que le LLM, grâce à ses données d'entraînement et à un prompt adapté, produise un GCS correctement instancié (càd. répondant à des critères de syntaxe, de cohérence sémantique et d'utilisabilité).

Problématique – Forces et faiblesses des LLMs pour produire des GCS

LLMs et GCS Peu de travaux scientifiques abordent directement le sujet des GCS.

Text2Graph (Wang'25) variante "OIE-based" (Open Information Extraction) s'appuie uniquement sur le texte fourni en entrée pour extraire les entités et leurs relations ; "ontology-driven" fait intervenir un texte contraint par une ontologie associée.

Intuition Produire un GCS, c'est s'aligner sur les caractéristiques essentielles d'un domaine de discours, et donc sur une ontologie → solliciter un LLM pour générer un GCS à partir d'une ontologie et de consignes fournies en entrée.

Ontology2Graph – Evaluer la capacité des LLMs à produire des GCS

QR. 1 Capacité des LLMs à générer de grands graphes (en ordre et taille)

QR. 2 Cohérence de ces graphes avec l'ontologie en question et le domaine associé, notamment en ce qui concerne l'instanciation de classes d'objets sémantiquement valides.

Hypothèse L'ontologie fournit suffisamment d'informations pour que le LLM, grâce à ses données d'entraînement et à un prompt adapté, produise un GCS correctement instancié (càd. répondant à des critères de syntaxe, de cohérence sémantique et d'utilisabilité).

Problématique – Forces et faiblesses des LLMs pour produire des GCS

LLMs et GCS Peu de travaux scientifiques abordent directement le sujet des GCS.

Text2Graph (Wang'25) variante "OIE-based" (Open Information Extraction) s'appuie uniquement sur le texte fourni en entrée pour extraire les entités et leurs relations ; "ontology-driven" fait intervenir un texte contraint par une ontologie associée.

Intuition Produire un GCS, c'est s'aligner sur les caractéristiques essentielles d'un domaine de discours, et donc sur une ontologie → solliciter un LLM pour générer un GCS à partir d'une ontologie et de consignes fournies en entrée.

Ontology2Graph – Evaluer la capacité des LLMs à produire des GCS

QR. 1 Capacité des LLMs à générer de grands graphes (en ordre et taille)

QR. 2 Cohérence de ces graphes avec l'ontologie en question et le domaine associé, notamment en ce qui concerne l'instanciation de classes d'objets sémantiquement valides.

Hypothèse L'ontologie fournit suffisamment d'informations pour que le LLM, grâce à ses données d'entraînement et à un prompt adapté, produise un GCS correctement instancié (càd. répondant à des critères de syntaxe, de cohérence sémantique et d'utilisabilité).

Ontology2Graph – Génération de GCS par LLM

Modèle	Capacités	⌈token⌉ entrée	⌈token⌉ sortie
Gpt-4.1-nano Data cutoff : 06/2024	Function calling Vision	1 047 576	32 768
Gemini-2.5-flash Data cutoff : 01/2025	Function calling Vision Web search Url context Reasoning	1 048 576	65 565

LLMs Utilisation de modèles à l'état de l'art en 2025.

Prompt Stratégie de prompt standard en première approche (description de tâche + instructions + contexte).

Ontologie NORIA-O (Tailhardat'24-2) : topologie et opérations réseaux telco.

Optim. Sélection de modèle (dont paramètres) et de prompt s.c. max(ordre graphe) et min(erreurs).

P1 (prompt initial, zero-shot)

```
Given a turtle schema (noria.ttl) describing classes and relations, provide a Turtle result
file that describes a new knowledge graph based on this specific schema. You have to create
Nodes linked together via the Relations described in the schema provided.
Dismiss any isolated nodes, each nodes must have at least one relation with another node.
Entities must come from example.com declared as prefix. Group Entities declaration and
Relationships in the same statement and use rdfs:label and rdfs:comment to provide more
clarity on Entities.
Never repeat the same information and respect RDF, RDFS, OWL vocabularies and Turtle syntax.
Never put explanations at the end of the file. Each prefix used in the nodes declarations
must be declared at the beginging of the turtle result file. Be creative.
```

Ontology2Graph – Génération de GCS par LLM

Modèle	Capacités	⌈token⌉ entrée	⌈token⌉ sortie
Gpt-4.1-nano Data cutoff : 06/2024	Function calling Vision	1 047 576	32 768
Gemini-2.5-flash Data cutoff : 01/2025	Function calling Vision Web search Url context Reasoning	1 048 576	65 565

LLMs Utilisation de modèles à l'état de l'art en 2025.

Prompt Stratégie de prompt standard en première approche (description de tâche + instructions + contexte).

Ontologie NORIA-O (Tailhardat'24-2) : topologie et opérations réseaux telco.

Optim. Sélection de modèle (dont paramètres) et de prompt s.c. max(ordre graphe) et min(erreurs).

P1 (prompt initial, zero-shot)

```
Given a turtle schema (noria.ttl) describing classes and relations, provide a Turtle result
file that describes a new knowledge graph based on this specific schema. You have to create
Nodes linked together via the Relations described in the schema provided.
Dismiss any isolated nodes, each nodes must have at least one relation with another node.
Entities must come from example.com declared as prefix. Group Entities declaration and
Relationships in the same statement and use rdfs:label and rdfs:comment to provide more
clarity on Entities.
Never repeat the same information and respect RDF, RDFS, OWL vocabularies and Turtle syntax.
Never put explanations at the end of the file. Each prefix used in the nodes declarations
must be declared at the beginging of the turtle result file. Be creative.
```

Ontology2Graph – Génération de GCS par LLM

Modèle	Capacités	⌈token⌉ entrée	⌈token⌉ sortie
Gpt-4.1-nano Data cutoff : 06/2024	Function calling Vision	1 047 576	32 768
Gemini-2.5-flash Data cutoff : 01/2025	Function calling Vision Web search Url context Reasoning	1 048 576	65 565

LLMs Utilisation de modèles à l'état de l'art en 2025.

Prompt Stratégie de prompt standard en première approche (description de tâche + instructions + contexte).

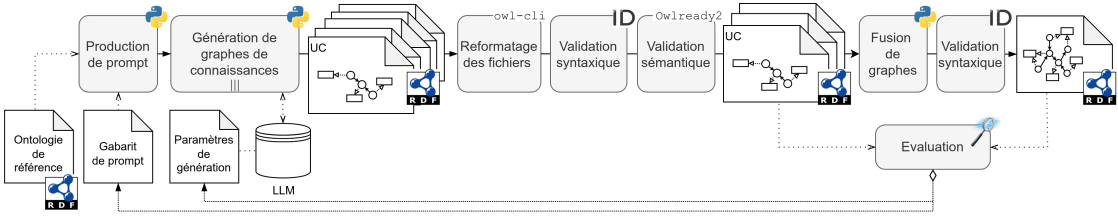
Ontologie NORIA-O (Tailhardat'24-2) : topologie et opérations réseaux telco.

Optim. Sélection de modèle (dont paramètres) et de prompt s.c. max(ordre graphe) et min(erreurs).

P1 (prompt initial, zero-shot)

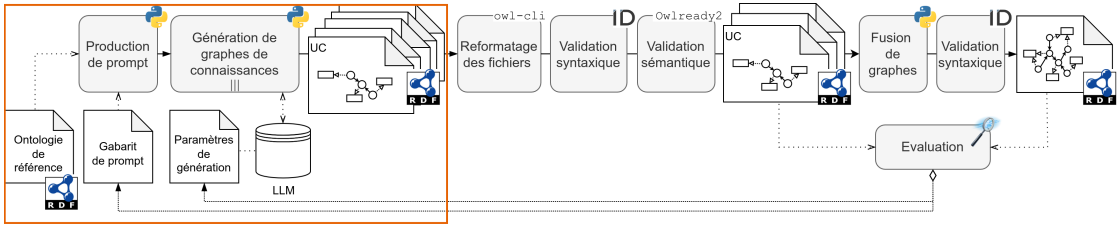
```
Given a turtle schema (noria.ttl) describing classes and relations, provide a Turtle result
file that describes a new knowledge graph based on this specific schema. You have to create
Nodes linked together via the Relations described in the schema provided.
Dismiss any isolated nodes, each nodes must have at least one relation with another node.
Entities must come from example.com declared as prefix. Group Entities declaration and
Relationships in the same statement and use rdfs:label and rdfs:comment to provide more
clarity on Entities.
Never repeat the same information and respect RDF, RDFS, OWL vocabularies and Turtle syntax.
Never put explanations at the end of the file. Each prefix used in the nodes declarations
must be declared at the beginging of the turtle result file. Be creative.
```

Ontology2Graph – Architecture fonctionnelle



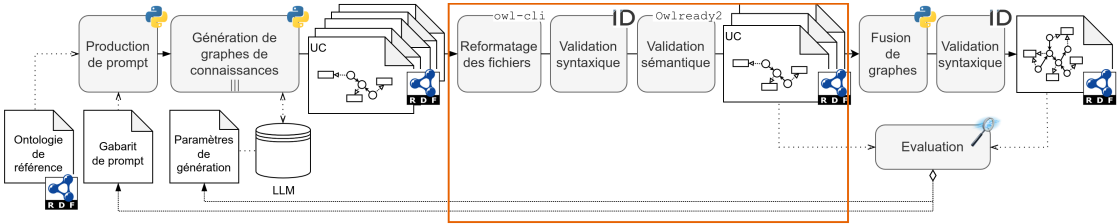
1. Production de n GCS à partir d'une ontologie de référence
2. Validation syntaxique et sémantique des n GCS (turtle-validator, Pellet, HermiT)
3. Fusion contrôlée pour obtenir un grand graphe (dépasser la limite de tokens des LLMs, discuté ci-après)
4. Identifier les modèles et les prompts pertinents (évaluation quanti. & quali., discutée ensuite)

Ontology2Graph – Architecture fonctionnelle



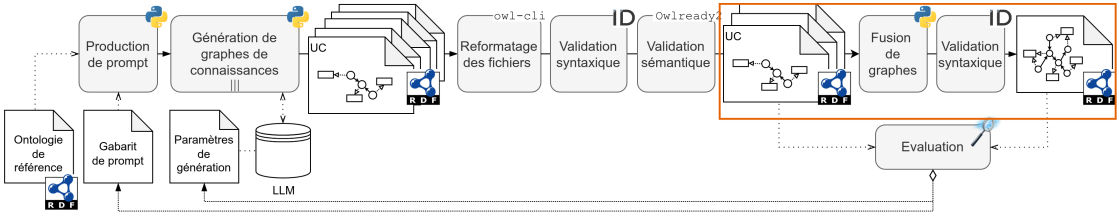
1. Production de n GCS à partir d'une ontologie de référence
2. Validation syntaxique et sémantique des n GCS (turtle-validator, Pellet, HermiT)
3. Fusion contrôlée pour obtenir un grand graphe (dépasser la limite de tokens des LLMs, discuté ci-après)
4. Identifier les modèles et les prompts pertinents (évaluation quanti. & quali., discutée ensuite)

Ontology2Graph – Architecture fonctionnelle



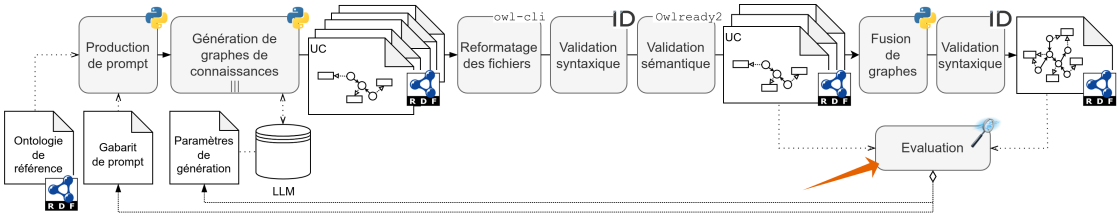
1. Production de n GCS à partir d'une ontologie de référence
2. Validation syntaxique et sémantique des n GCS (turtle-validator, Pellet, HermiT)
3. Fusion contrôlée pour obtenir un grand graphe (dépasser la limite de tokens des LLMs, discuté ci-après)
4. Identifier les modèles et les prompts pertinents (évaluation quanti. & quali., discutée ensuite)

Ontology2Graph – Architecture fonctionnelle



1. Production de n GCS à partir d'une ontologie de référence
2. Validation syntaxique et sémantique des n GCS (turtle-validator, Pellet, HermiT)
3. Fusion contrôlée pour obtenir un grand graphe (dépasser la limite de tokens des LLMs, discuté ci-après)
4. Identifier les modèles et les prompts pertinents (évaluation quanti. & quali., discutée ensuite)

Ontology2Graph – Architecture fonctionnelle



1. Production de n GCS à partir d'une ontologie de référence
2. Validation syntaxique et sémantique des n GCS (turtle-validator, Pellet, HermiT)
3. Fusion contrôlée pour obtenir un grand graphe (dépasser la limite de tokens des LLMs, discuté ci-après)
4. Identifier les modèles et les prompts pertinents (évaluation quanti. & quali., discutée ensuite)

Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traitement Nœuds homonymes $\simeq$ correspondance exacte entre GCS

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.

Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

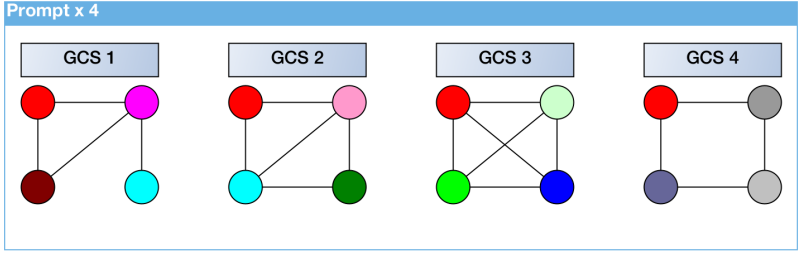
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traitement Nœuds homonymes $\simeq$ correspondance exacte entre GCS

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

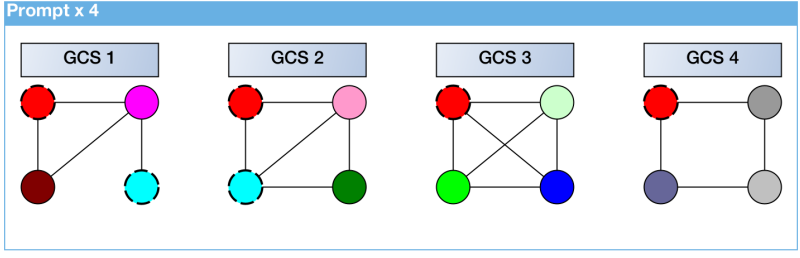
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traîtement Nœuds homonymes $\simeq$ correspondance exacte entre GCS → dépasser la limite $\lceil$token sortie$\rceil$ en post-génération par assemblage des GCS (fichiers Turtle) !

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

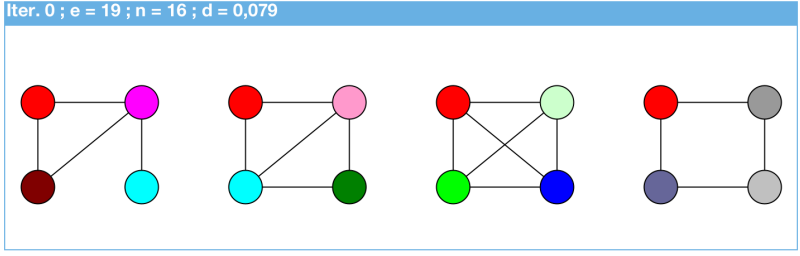
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traîtement Nœuds homonymes $\simeq$ correspondance exacte entre GCS $\rightarrow$ dépasser la limite $\lceil$token sortie$\rceil$ en post-génération par assemblage des GCS (fichiers Turtle) !

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

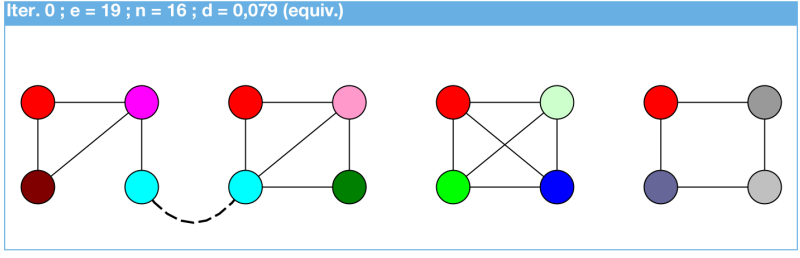
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traîtement Nœuds homonymes $\simeq$ correspondance exacte entre GCS $\rightarrow$ dépasser la limite $\lceil$token sortie$\rceil$ en post-génération par assemblage des GCS (fichiers Turtle) !

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

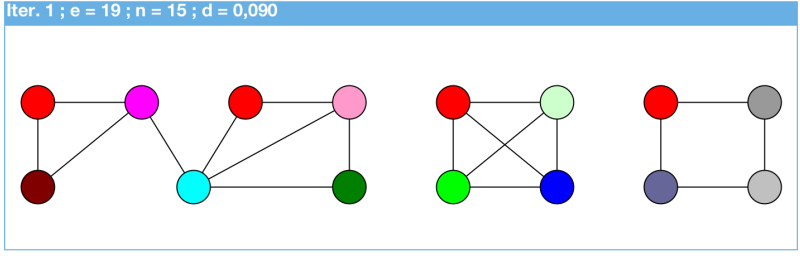
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil \text{token sortie} \rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traîtement Nœuds homonymes $\simeq$ correspondance exacte entre GCS $\rightarrow$ dépasser la limite $\lceil \text{token sortie} \rceil$ en post-génération par assemblage des GCS (fichiers Turtle) !

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

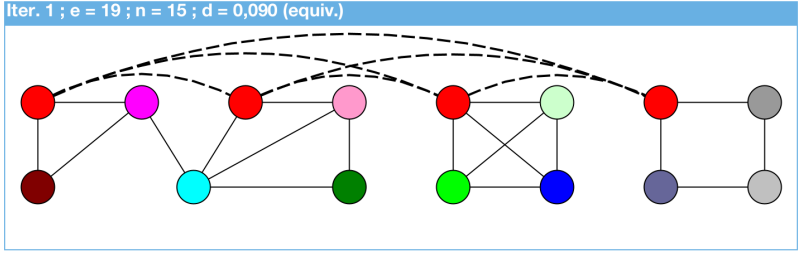
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traîtement Nœuds homonymes $\simeq$ correspondance exacte entre GCS $\rightarrow$ dépasser la limite $\lceil$token sortie$\rceil$ en post-génération par assemblage des GCS (fichiers Turtle) !

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

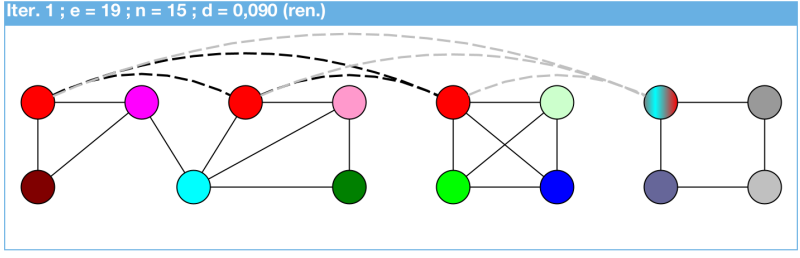
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traîtement Nœuds homonymes $\simeq$ correspondance exacte entre GCS $\rightarrow$ dépasser la limite $\lceil$token sortie$\rceil$ en post-génération par assemblage des GCS (fichiers Turtle) !

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Ontology2Graph – Fusion de graphes par égalité stricte de nom des entités

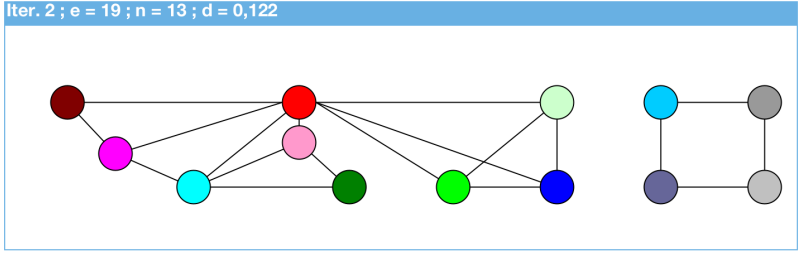
Génération Pour un appel au LLM, l'ordre et la taille du GCS résultant est limité par $\lceil$token sortie$\rceil$,

p.ex. rel./fichier $\sim \frac{\lceil \text{token sortie} \rceil \times \text{car./token}}{\text{car./rel.}} = \frac{65565 \times 4}{100} = 2622$ (Gemini-2.5-flash)

Stabilité Les appels répétés à P1 révèlent la présence de nœuds homonymes (càd. identiques de par leur dénomination) dans les séries de graphes générés.

Traîtement Nœuds homonymes $\simeq$ correspondance exacte entre GCS $\rightarrow$ dépasser la limite $\lceil$token sortie$\rceil$ en post-génération par assemblage des GCS (fichiers Turtle) !

Fusion contrôlée Le degré du GCS est contrôlé par renommage ciblé des nœuds homonymes durant la concaténation.



Evaluation – Performance des modèles avec le prompt P1

T	top_p	F_{err}	$\overline{\text{token sortie}}$	$\overline{n}$	$\overline{\Delta(\mathcal{G})}$
Gpt-4.1-nano					
0,1	0,1	7	1952	20	0,0620
0,1	0,5	7	2128	23	0,0493
0,1	1,0	9	2548	**37**	0,0292
0,5	0,1	8	2038	22	0,0545
0,5	0,5	8	2198	20	0,0515
0,5	1,0	7	1923	25	0,0494
1,0	0,1	9	**5483**	19	0,0760
1,0	0,5	**5**	1930	22	0,0665
1,0	1,0	8	1574	12	0,1023
Gemini-2.5-flash					
0,1	0,1	**0**	**16 248**	63	0,0230
0,1	0,5	7	9246	50	0,0304
0,1	1,0	7	11 867	**78**	0,0216
0,5	0,1	**0**	15 883	63	0,0230
0,5	0,5	4	12 181	58	0,0254
0,5	1,0	7	11 123	55	0,0296
1,0	0,1	**0**	15 882	63	0,0230
1,0	0,5	7	11 811	66	0,0215
1,0	1,0	5	11 300	68	0,0208

Objectif Identifier l'influence des paramètres relatifs à la créativité des modèles sur la qualité des graphes produits en termes de nombre de nœuds par graphe et de graphes syntaxiquement corrects.

Protocole Création de dix graphes consécutifs par modèle ; variation des paramètres T (température) et top_p (nucleus sampling).

Observations

- $\overline{\text{token sortie}} << [\text{token sortie}]$,
- Pas de corrélation détectée entre la variation des paramètres de créativité et la qualité des graphes,
- Gemini-2.5-flash: génère systématiquement plus de nœuds et de graphes valides,
- Gemini-2.5-flash: diversité satisfaisante de fichiers (dix fichiers différents) avec top_p $\geq 0,4$.

$\hookrightarrow$ Gemini-2.5-flash sélectionné pour les expérimentations suivantes.

Evaluation – Performance des modèles avec le prompt P1

T	top_p	F_{err}	$\overline{\text{token sortie}}$	$\overline{n}$	$\overline{\Delta(\mathcal{G})}$
			Gpt-4.1-nano		
0,1	0,1	7	1952	20	0,0620
0,1	0,5	7	2128	23	0,0493
0,1	1,0	9	2548	**37**	0,0292
0,5	0,1	8	2038	22	0,0545
0,5	0,5	8	2198	20	0,0515
0,5	1,0	7	1923	25	0,0494
1,0	0,1	9	**5483**	19	0,0760
1,0	0,5	**5**	1930	22	0,0665
1,0	1,0	8	1574	12	0,1023
			Gemini-2.5-flash		
0,1	0,1	**0**	**16 248**	63	0,0230
0,1	0,5	7	9246	50	0,0304
0,1	1,0	7	11 867	**78**	0,0216
0,5	0,1	**0**	15 883	63	0,0230
0,5	0,5	4	12 181	58	0,0254
0,5	1,0	7	11 123	55	0,0296
1,0	0,1	**0**	15 882	63	0,0230
1,0	0,5	7	11 811	66	0,0215
1,0	1,0	5	11 300	68	0,0208

Objectif Identifier l'influence des paramètres relatifs à la créativité des modèles sur la qualité des graphes produits en termes de nombre de nœuds par graphe et de graphes syntaxiquement corrects.

Protocole Création de dix graphes consécutifs par modèle ; variation des paramètres T (température) et top_p (nucleus sampling).

Observations

- $\overline{\text{token sortie}} << \lceil \text{token sortie} \rceil$,
- Pas de corrélation détectée entre la variation des paramètres de créativité et la qualité des graphes,
- Gemini-2.5-flash: génère systématiquement plus de nœuds et de graphes valides,
- Gemini-2.5-flash: diversité satisfaisante de fichiers (dix fichiers différents) avec top_p $\geq 0,4$.

↪ Gemini-2.5-flash sélectionné pour les expérimentations suivantes.

Evaluation – Optimisation du prompt pour Gemini-2.5-flash

Prompt	Créé manuellement	Optimisé manuellement		Optimisé par IA		Créé par IA
P1	✓	✗		✗		✗
P2	✗	✓	(P1)	✗		✗
P2_1	✗	✓	(P2)	✗		✗
P3	✗	✗		✓	(P2)	✗
P3_1	✗	✗		✓	(P2_1)	✗
P4	✗	✗		✗		✓
P4_1	✗	✓	(P4)	✗		✗

Prompt	$\overline{\text{token raisonnement}}$	$\overline{\text{token texte}}$	$\overline{\text{token sortie}}$	$\overline{n}$	F_{err}
P1	13 033	11 729	24 781	94	15
P2	17 926	25 013	**42 940**	115	14
P2_1	14 275	13 487	27 756	99	8
P3	**23 839**	17 168	38 657	120	13
P3_1	19 397	21 911	41 325	123	15
P4	8 416	21 870	30 236	128	11
P4_1	10 260	**26 624**	36 885	**159**	6
P4_1*	2 232	16 317	18 559	94	11
P4_1	4 619	19 725	24 345	144	**4**
P4_1*	3 186	13 044	16 731	102	11
P4_1	6 503	19 076	25 579	134	6
P4_1*	6 446	11 564	18 010	106	12

Prompt système “using near 60 000 output tokens to maximize your response” (sauf *)

Objectif Identifier les caractéristiques du prompt minimisant les principales erreurs détectées et maximisant l’ordre du graphe.

Protocole Essai-analyse sur la structure et les consignes ; génération de 20 graphes prompt ; $T = 0{,}1$ et top_p $= 0{,}4$.

Observations

- $\overline{\text{token sortie}} << \overline{\text{token sortie}}$, malgré le prompt système,
- Meilleurs résultats obtenus avec P4_1,
- Prompt système pour maximiser $\overline{\text{token sortie}}$: la consigne ne permet pas de s’approcher, mais l’enlever dégrade $\overline{n}$ et F_{err}.

↪ Meilleure stratégie : demander à un LLM de générer un prompt + optimisation manuelle a posteriori + prompt système.

Evaluation – Optimisation du prompt pour Gemini-2.5-flash

Prompt	Créé manuellement	Optimisé manuellement		Optimisé par IA		Créé par IA
P1	✓	✗		✗		✗
P2	✗	✓	(P1)	✗		✗
P2_1	✗	✓	(P2)	✗		✗
P3	✗	✗		✓	(P2)	✗
P3_1	✗	✗		✓	(P2_1)	✗
P4	✗	✗		✗		✓
P4_1	✗	✓	(P4)	✗		✗

Prompt	$\overline{\text{token raisonnement}}$	$\overline{\text{token texte}}$	$\overline{\text{token sortie}}$	$\overline{n}$	F_{err}
P1	13 033	11 729	24 781	94	15
P2	17 926	25 013	**42 940**	115	14
P2_1	14 275	13 487	27 756	99	8
P3	**23 839**	17 168	38 657	120	13
P3_1	19 397	21 911	41 325	123	15
P4	8 416	21 870	30 236	128	11
P4_1	10 260	**26 624**	36 885	**159**	6
P4_1*	2 232	16 317	18 559	94	11
P4_1	4 619	19 725	24 345	144	**4**
P4_1*	3 186	13 044	16 731	102	11
P4_1	6 503	19 076	25 579	134	6
P4_1*	6 446	11 564	18 010	106	12

Prompt système “using near 60 000 output tokens to maximize your response” (sauf *)

Objectif Identifier les caractéristiques du prompt minimisant les principales erreurs détectées et maximisant l’ordre du graphe.

Protocole Essai-analyse sur la structure et les consignes ; génération de 20 graphes prompt ; T $= 0{,}1$ et top_p $= 0{,}4$.

Observations

- $\overline{\text{token sortie}} << \lceil \text{token sortie} \rceil$, malgré le prompt système,
- Meilleurs résultats obtenus avec P4_1,
- Prompt système pour maximiser $\overline{\text{token sortie}}$: la consigne ne permet pas de s’approcher, mais l’enlever dégrade $\overline{n}$ et F_{err}.

↪ Meilleure stratégie : demander à un LLM de générer un prompt + optimisation manuelle a posteriori + prompt système.

Evaluation – Optimisation à l'aide d'un prompt de création de prompts

P4_1 (GPT-4.1-mini, prompt dérivé de P4 en supprimant toute les références au préfixe `ldp`)

```
Tu es un expert en modélisation de graphes de connaissances RDF/OWL et en syntaxe Turtle. Ta mission est de générer un
graphe de connaissances complet, dense, conforme et connecté, en utilisant exclusivement le vocabulaire NORIA défini
ci-dessous. Consignes précises :

1. Respect strict du vocabulaire NORIA : Utilise uniquement les classes, propriétés, préfixes et concepts définis dans le
vocabulaire NORIA fourni. Ne crée aucune entité, propriété ou préfixe hors de ce vocabulaire.
2. Conformité syntaxique parfaite : Le graphe doit être valide en syntaxe Turtle, sans erreurs ni entités tronquées. Chaque
triplet doit être complet et correctement formé.
3. Complétude des entités : Pour chaque instance créée, fournis un ensemble complet de descripteurs pertinents selon le
vocabulaire NORIA, notamment : rdf:type (classe exacte), rdfs:label en anglais (et en Français si possible), rdfs:comment ou
skos:example pour enrichir la description, relations avec d'autres entités via les propriétés du vocabulaire (ex :
noria:resourceManagedBy, noria:applicationModuleOf, etc.)
4. Interdiction du préfixe ldp : le préfixe ldp (Linked Data Platform) ne doit pas être utilisé, ni déclaré, ni référencé
dans le graphe.
5. Maximisation du nombre d'entités : Génère autant d'entités (nœuds) que possible dans la limite des tokens autorisés, en
créant un graphe riche, varié et détaillé.
6. Garanties de connexité et densité : Chaque entité doit être liée à au moins une autre entité via une propriété du
vocabulaire NORIA. Le graphe doit présenter une densité raisonnable, avec de nombreuses relations entre entités, reflétant
des cas réels d'infrastructures IT et de gestion d'incidents. Évite les entités isolées ou orphelines.
7. Organisation claire : Commence par déclarer tous les préfixes NORIA et standards (rdf, rdfs, owl, xsd, foaf, etc.) tels
que définis dans le vocabulaire. Organise la sortie en blocs logiques par type d'entité (ex : Resources, Applications,
TroubleTickets, EventRecords, etc.)

Vocabulaire NORIA (extraits clés) : Classes principales : noria:Resource, noria:Application, [...], etc. Propriétés
importantes : noria:resourceManagedBy (Resource → foaf:Agent), [...], etc.

Utilise les annotations rdfs:label, rdfs:comment, skos:example pour enrichir les entités.
```

Evaluation – Analyse qualitative par des raisonneurs automatiques

Objectif Vérifier la cohérence des GCS à l'ontologie de référence (NORIA-O).

Protocole Appel des raisonneurs HermiT et Pellet (via owlready2) ; modes `OwlReady Inconsistent Ontology Error` et `Default world inconsistent classes` ; analyse des logs d'exécution et d'erreurs.

Observations

- Aucune classe incohérente détectée dans le graphe de données.

↪ Conformité sémantique globale sous réserve d'un post-traitement.

↪ Quid de la représentativité du GCS à un domaine ? ...

Ontology Inconsistante (exemple)

`noria:documentStatusHistory rdfs:range noria:ChangeRequest, noria:TroubleTicket`

Les classes `noria:ChangeRequest` et `noria:TroubleTicket` sont disjointes → contradiction logique.

DatatypeProperty (exemple)

`noria:troubleTicketSupervisionTool "Nagios"@en` ≠ `rdfs:range xsd:string` (`rdfs:range xsd:langString` en RDF1.1)

Littéraux non conformes (exemple)

`noria:networkInterfaceLaserRxHighPower-WarningThreshold "12.5"` ≠ `rdfs:range xsd:int`

Evaluation – Analyse qualitative par des raisonneurs automatiques

Objectif Vérifier la cohérence des GCS à l'ontologie de référence (NORIA-O).

Protocole Appel des raisonneurs HermiT et Pellet (via owlready2) ; modes `OwlReady Inconsistent Ontology Error` et `Default world inconsistent classes` ; analyse des logs d'exécution et d'erreurs.

Observations

- Aucune classe incohérente détectée dans le graphe de données.
- `OwlReady Inconsistent Ontology Error` (cause NORIA-O).
- Instanciation incorrecte de relations DatatypeProperty (cause LLM).
- Littéraux non conformes au type attendu (cause LLM).

Ontology Inconsistante (exemple)

`noria:documentStatusHistory rdfs:range noria:ChangeRequest, noria:TroubleTicket`

Les classes `noria:ChangeRequest` et `noria:TroubleTicket` sont disjointes → contradiction logique.

DatatypeProperty (exemple)

`noria:troubleTicketSupervisionTool "Nagios"@en` ≠ `rdfs:range xsd:string` (`rdfs:range xsd:langString` en RDF1.1)

Littéraux non conformes (exemple)

`noria:networkInterfaceLaserRxHighPowerWarningThreshold "12.5"` ≠ `rdfs:range xsd:int`

↪ Conformité sémantique globale sous réserve d'un post-traitement.

↪ Quid de la représentativité du GCS à un domaine ? ...

Evaluation – Analyse qualitative par des raisonneurs automatiques

Objectif Vérifier la cohérence des GCS à l'ontologie de référence (NORIA-O).

Protocole Appel des raisonneurs HermiT et Pellet (via owlready2) ; modes `OwlReady Inconsistent Ontology Error` et `Default world inconsistent classes` ; analyse des logs d'exécution et d'erreurs.

Observations

- Aucune classe incohérente détectée dans le graphe de données.
- `OwlReady Inconsistent Ontology Error` (cause NORIA-O).
- Instanciation incorrecte de relations DatatypeProperty (cause LLM).
- Littéraux non conformes au type attendu (cause LLM).

Ontology Inconsistante (exemple)

`noria:documentStatusHistory rdfs:range noria:ChangeRequest, noria:TroubleTicket`

Les classes `noria:ChangeRequest` et `noria:TroubleTicket` sont disjointes → contradiction logique.

DatatypeProperty (exemple)

`noria:troubleTicketSupervisionTool "Nagios"@en` ≠ `rdfs:range xsd:string` (`rdfs:range xsd:langString` en RDF1.1)

Littéraux non conformes (exemple)

`noria:networkInterfaceLaserRxHighPowerWarningThreshold "12.5"` ≠ `rdfs:range xsd:int`

↪ Conformité sémantique globale sous réserve d'un post-traitement.

↪ Quid de la représentativité du GCS à un domaine ? ...

Evaluation – Analyse qualitative par des raisonneurs automatiques

Objectif Vérifier la cohérence des GCS à l'ontologie de référence (NORIA-O).

Protocole Appel des raisonneurs HermiT et Pellet (via owlready2) ; modes `OwlReady Inconsistent Ontology Error` et `Default world inconsistent classes` ; analyse des logs d'exécution et d'erreurs.

Observations

- Aucune classe incohérente détectée dans le graphe de données.
- `OwlReady Inconsistent Ontology Error` (cause NORIA-O).
- Instanciation incorrecte de relations DatatypeProperty (cause LLM).
- Littéraux non conformes au type attendu (cause LLM).

Ontology Inconsistante (exemple)

`noria:documentStatusHistory rdfs:range noria:ChangeRequest, noria:TroubleTicket`

Les classes `noria:ChangeRequest` et `noria:TroubleTicket` sont disjointes → contradiction logique.

DatatypeProperty (exemple)

`noria:troubleTicketSupervisionTool "Nagios"@en` ≠ `rdfs:range xsd:string`
(`rdfs:range xsd:langString` en RDF1.1)

Littéraux non conformes (exemple)

`noria:networkInterfaceLaserRxHighPowerWarningThreshold "12.5"` ≠ `rdfs:range xsd:int`

↪ Conformité sémantique globale sous réserve d'un post-traitement.

↪ Quid de la représentativité du GCS à un domaine ? ...

Evaluation – Analyse qualitative par des raisonneurs automatiques

Objectif Vérifier la cohérence des GCS à l'ontologie de référence (NORIA-O).

Protocole Appel des raisonneurs HermiT et Pellet (via owlready2) ; modes `OwlReady Inconsistent Ontology Error` et `Default world inconsistent classes` ; analyse des logs d'exécution et d'erreurs.

Observations

- Aucune classe incohérente détectée dans le graphe de données.
- `OwlReady Inconsistent Ontology Error` (cause NORIA-O).
- Instanciation incorrecte de relations DatatypeProperty (cause LLM).
- Littéraux non conformes au type attendu (cause LLM).

Ontology Inconsistante (exemple)

`noria:documentStatusHistory rdfs:range noria:ChangeRequest, noria:TroubleTicket`

Les classes `noria:ChangeRequest` et `noria:TroubleTicket` sont disjointes → contradiction logique.

DatatypeProperty (exemple)

`noria:troubleTicketSupervisionTool "Nagios"@en` ≠ `rdfs:range xsd:string` (`rdfs:range xsd:langString` en RDF1.1)

Littéraux non conformes (exemple)

`noria:networkInterfaceLaserRxHighPower-WarningThreshold "12.5"` ≠ `rdfs:range xsd:int`

↪ Conformité sémantique globale sous réserve d'un post-traitement.

↪ Quid de la représentativité du GCS à un domaine ? ...

Evaluation – Analyse qualitative par des raisonneurs automatiques

Objectif Vérifier la cohérence des GCS à l'ontologie de référence (NORIA-O).

Protocole Appel des raisonneurs HermiT et Pellet (via owlready2) ; modes `OwlReady Inconsistent Ontology Error` et `Default world inconsistent classes` ; analyse des logs d'exécution et d'erreurs.

Observations

- Aucune classe incohérente détectée dans le graphe de données.
- `OwlReady Inconsistent Ontology Error` (cause NORIA-O).
- Instanciation incorrecte de relations DatatypeProperty (cause LLM).
- Littéraux non conformes au type attendu (cause LLM).

Ontology Inconsistante (exemple)

`noria:documentStatusHistory rdfs:range noria:ChangeRequest, noria:TroubleTicket`

Les classes `noria:ChangeRequest` et `noria:TroubleTicket` sont disjointes → contradiction logique.

DatatypeProperty (exemple)

`noria:troubleTicketSupervisionTool "Nagios"@en` ≠ `rdfs:range xsd:string` (`rdfs:range xsd:langString` en RDF1.1)

Littéraux non conformes (exemple)

`noria:networkInterfaceLaserRxHighPower-WarningThreshold "12.5"` ≠ `rdfs:range xsd:int`

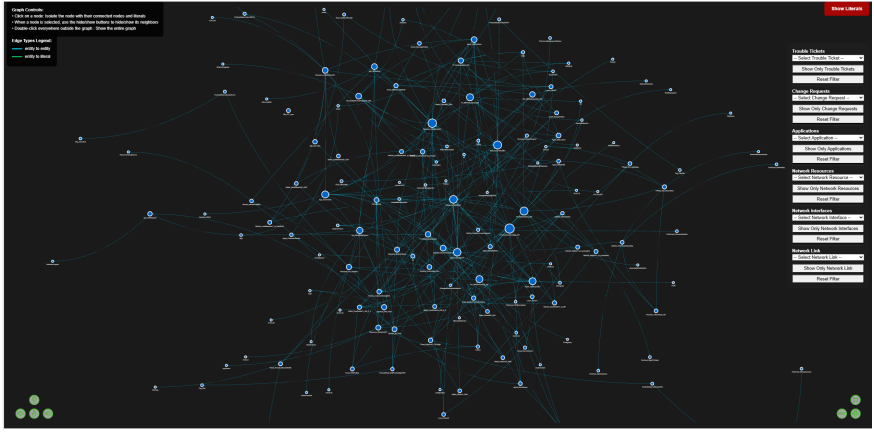
↪ Conformité sémantique globale sous réserve d'un post-traitement.

↪ Quid de la représentativité du GCS à un domaine ? ...

Evaluation – Analyse qualitative par un panel d'experts (1/2)

Objectif Vérifier la pertinence des GCS pour le domaine d'application (exploitation des réseaux / NORIA-O).

Protocole Analyse critique d'un GCS (post fusion) ; six experts du Groupe Orange ; application Web (navigation et filtrage du GCS) ; questionnaire (niveau de pertinence des relations et litéraux + commentaires libres).



Evaluation – Analyse qualitative par un panel d'experts (2/2)

Entités	Pertinence des relations entre nœuds					Pertinence des littéraux				
	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆	n.s.p.	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆	n.s.p.
noria:TroubleTicket	**50%**	33%	0%	0%	17%	**66%**	17%	0%	0%	17%
noria:ChangeRequest	**66%**	17%	0%	0%	17%	**66%**	17%	0%	0%	17%
noria:Application	**50%**	17%	17%	0%	17%	**66%**	17%	0%	0%	17%
noria:NetworkResource	33%	**50%**	0%	0%	17%	**66%**	0%	17%	0%	17%
noria:NetworkInterface	17%	**33%**	**33%**	0%	17%	**50%**	0%	33%	0%	17%
noria:NetworkLink	**33%**	17%	**33%**	0%	17%	**66%**	0%	0%	0%	33%

⋆⋆⋆⋆ = toutes les relations/littéraux sont pertinent(e)s ; ⋆⋆⋆= les relations/littéraux sont pertinent(e)s mais il manque des informations importantes ; ⋆⋆ = certain(e)s relations/littéraux sont pertinent(e)s mais certain(e)s sont inutiles ou incohérentes ; ⋆ = toutes les relations/littéraux sont inutiles ou incohérent(e)s ; n.s.p. = ne se prononce pas.

Observations

- ObjectProperty (entité-entité) : $42\,\%$ d'avis de niveau maximal (⋆⋆⋆⋆)
- DatatypeProperty (entité-litéral) : $64\,\%$ de niveau maximal.
- Facette structurelle des réseaux (`noria:NetworkResource`, `noria:NetworkInterface` et `noria:NetworkLink`) : satisfaction moindre (⋆⋆⋆et ⋆⋆).
- Commentaires sur l'absence de "réseau multicouche", de "distinction entre port physique et logique", de "littéraux relatifs au release management des applications et à la résolution d'incidents".

↪ Avis globalement très favorable sur le GCS.

↪ Manque d'ancrage du LLM sur des situations réelles, représentables par l'ontologie de référence (NORIA-O) mais dont l'inférence n'est pas du rôle de l'ontologie elle-même.

Evaluation – Analyse qualitative par un panel d'experts (2/2)

Entités	Pertinence des relations entre nœuds					Pertinence des littéraux				
	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆	n.s.p.	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆	n.s.p.
noria:TroubleTicket	**50%**	33%	0%	0%	17%	**66%**	17%	0%	0%	17%
noria:ChangeRequest	**66%**	17%	0%	0%	17%	**66%**	17%	0%	0%	17%
noria:Application	**50%**	17%	17%	0%	17%	**66%**	17%	0%	0%	17%
noria:NetworkResource	33%	**50%**	0%	0%	17%	**66%**	0%	17%	0%	17%
noria:NetworkInterface	17%	**33%**	**33%**	0%	17%	**50%**	0%	33%	0%	17%
noria:NetworkLink	**33%**	17%	**33%**	0%	17%	**66%**	0%	0%	0%	33%

⋆⋆⋆⋆ = toutes les relations/littéraux sont pertinent(e)s ; ⋆⋆⋆= les relations/littéraux sont pertinent(e)s mais il manque des informations importantes ; ⋆⋆ = certain(e)s relations/littéraux sont pertinent(e)s mais certain(e)s sont inutiles ou incohérentes ; ⋆ = toutes les relations/littéraux sont inutiles ou incohérent(e)s ; n.s.p. = ne se prononce pas.

Observations

- ObjectProperty (entité-entité) : $42\,\%$ d'avis de niveau maximal (⋆⋆⋆⋆)
- DatatypeProperty (entité-litéral) : $64\,\%$ de niveau maximal.
- Facette structurelle des réseaux (`noria:NetworkResource`, `noria:NetworkInterface` et `noria:NetworkLink`) : satisfaction moindre (⋆⋆⋆et ⋆⋆).
- Commentaires sur l'absence de "réseau multicouche", de "distinction entre port physique et logique", de "littéraux relatifs au release management des applications et à la résolution d'incidents".

↪ Avis globalement très favorable sur le GCS.

↪ Manque d'ancrage du LLM sur des situations réelles, représentables par l'ontologie de référence (NORIA-O) mais dont l'inférence n'est pas du rôle de l'ontologie elle-même.

Résumé et perspectives

Problème	Evaluer la capacité des LLMs à produire des GCS grands et pertinents pour un domaine d'application donné.
Hypothèse	Ontologie + données d'entraînement du LLM + prompt adapté → GCS correctement instancié.
Approche	Ontology2Graph : prompt engineering + validation syntaxique et sémantique + fusion de graphes + panel d'experts.
Résultats	Importance du modèle et du prompt pour max(ordre graphe) et min(erreurs) ; nécessité d'un post-traîtement (validation, fusion) ; manque d'ancrage du LLM sur des situations réelles dont l'inférence n'est pas du rôle de l'ontologie elle-même.
Travaux futurs	Autres LLMs ; autres ontologies et validation systématique avec CQs ; prompt avec consignes suplémentaires (besoins métiers spécifiques) ; stratégies de fusion (contrôle de la densité, propriétés redondantes ou conflictuelles).

Publication – IC @ PFIA 2026

Philippe Dooze, Lionel Tailhardat, Ovidiu Pascal.
Sur la génération de grands graphes de connaissances synthétiques à l'aide de modèles de langage (LLM) via une approche Ontology2Graph.

https://github.com/Orange-OpenSource/Ontology2Graph

Annexes

Références bibliographiques

(Bufort'25) A. Bufort, L. Tailhardat. "FGdiffusion : Large-Scale Knowledge Graph Generation Using a Diffusion Approach", 2025.

(Dessi'25) D. Dessí, et al. "CS-KG 2.0: A Large-scale Knowledge Graph of Computer Science", 2025.

(Fu'25) S. Fu, et al. "Utilizing Language Models For Synthetic Knowledge Graph Generation", 2025.

(Francart'14) T. Francart, et al. https://www.sparna.fr/fr/references/inra/, 2014.

(James'24) James S., et al. "SAKURA : Synthetic Cyber Knowledge Graph", 2024.

(Nikolentzos'23) G. Nikolentzos, et al. "Synthetic Electronic Health Records Generated with Variational Graph Autoencoders", 2023.

(Pascal'26) O. Pascal, et al. https://github.com/Orange-OpenSource/pygraft-gen, 2026.

(Tailhardat'24) L. Tailhardat. "Anomaly Detection using Knowledge Graphs and Synergistic Reasoning - Application to Network Management and Cyber Security", 2024.

(Tailhardat'24-2) L. Tailhardat, et al. "NORIA-O: An Ontology for Anomaly Detection and Incident Management in ICT Systems", 2024.

(Wang'25) D. Wang, M. I Waihara. "OSKGC: A Benchmark for Ontology Schema-based Knowledge Graph Construction from Text", 2025.

Evaluation – Optimisation de prompts

P2 (optimisation manuelle de P1, séquencement des tâches et explicitation des erreurs à éviter)

```
Given a turtle schema (Noria.ttl) describing classes and relations, provide a turtle result file that
describes a new synthetic knowledge graph based on this specific schema. You have to:
1) You must Respect RDF, RDFS, OWL vocabularies and TURTLE syntax.
2) Never start a statement with punctuation mark.
3) Never use ldp prefix.
4) Be careffull of syntax related to XMLSchema and xsd prefix each value must be surrounded by quote like
that: "value"^^xsd:type, if it is not the case, correct it.
5) Each prefix used must be declared at the begining of the turtle result file with @prefix.
6) The synthtetic knowledge graph you will provided must be as big as possible with a maximum number of
entities.
7) Create entities based on all the owl class declared in the provided schema.
8) Create entities that belong to the same owl class class declared in the provided schema.
9) Each node must be link to at least one another node with an ObjectProperties.
10) Use all the ObjectProperties declared in the provided schema to link the entities together.
11) Group entities declaration and their relationships in the same statement and Use rdfs:label and
rdfs:comment to provide more clarity on Entities.
12) Create literals on each entites created based on the DatatypeProperty declared in the provided schema.
Be creative.
```

Evaluation – Optimisation de prompts

P3 (optimisation par GPT-4.1-mini de P2)

```
Given a Turtle schema file (Noria.ttl) that defines OWL classes, ObjectProperties, and DatatypeProperties, generate a new synthetic knowledge graph as
a Turtle file that strictly conforms to RDF, RDFS, OWL vocabularies and Turtle syntax. Your output must satisfy all the following requirements:
1. **Syntax and Vocabulary Compliance:**
- The output must be a valid Turtle file, fully respecting RDF, RDFS, OWL vocabularies and Turtle syntax.
- Declare all necessary prefixes at the very beginning using `@prefix`.
- Use correct XMLSchema datatypes with the `xsd:` prefix; all literal values must be quoted and typed as `"value"^^xsd:type`. Correct any deviations.
2. **Graph Size and Coverage:**
- Generate the largest possible synthetic knowledge graph within token limits, maximizing the number of entities (nodes).
- Create at least one instance (individual) for **each OWL class** declared in the provided schema.
- For each entity, instantiate all applicable DatatypeProperties with appropriate literal values.
3. **Entity Linking:**
- Link every entity to at least one other entity using **all ObjectProperties** declared in the schema.
- Use ObjectProperties to create meaningful, non-trivial connections between entities, ensuring the graph is richly interconnected.
4. **Statements and Grouping:**
- Group entity declarations and their relationships within the same Turtle statement whenever possible, to improve readability and compactness.
- Add `rdfs:label` and `rdfs:comment` annotations to each entity to provide clear, human-readable descriptions.
5. **Output Formatting and Constraints:**
- The last entity statement must be complete and not truncated.
- Never start any Turtle statement with a punctuation mark.
- Do not use the `ldp` prefix anywhere in the output.
- Do not include any explanations, comments, or text outside the Turtle syntax in the output.
6. **Creativity and Realism:**
- Be creative in generating entity names, labels, comments, and literal values, while respecting the schema constraints.
- Ensure the synthetic graph is coherent and plausible as a knowledge graph.
---
**Important:** The output must be a single, valid Turtle file representing the synthetic knowledge graph based on the provided schema, ready for
direct use or validation.
---
Be thorough and precise.
```

Evaluation – Prompt d'optimisation de prompts

Prompt ayant mené à P3

```
Tu es un prompt engineer et tu dois optimiser un prompt visant la génération de
graphes de connaissances RDFS/OWL selon un vocabulaire fournis par l'utilisateur.

Optimise le prompt que je vais te donner de façon à maximiser la qualité du graphe
produit (p.ex. en terme de conformité syntaxique) et le nombre d'entités (nœuds)
produits (notamment en rapport à la limite du nombre de tokens qu'un LLM peut
produire).
```

Evaluation – Prompt de création de prompts

Prompt ayant mené à P4

```
Tu es un prompt engineer et tu dois produire un prompt qui vise la génération de
graphes de connaissances RDFS/OWL en syntaxe Turtle selon un vocabulaire fournit par
l'utilisateur.

Veille à ce que le prompt maximise la qualité du graphe produit (p.ex. en terme de
conformité syntaxique, de complétude des descripteurs de chaque entité, de respect
du vocabulaire de référence, absence d'entités tronquées en fin de résultat),
maximise le nombre d'entités (nœuds) produits (notamment en rapport à la limite du
nombre de tokens qu'un LLM peut produire), et présente des garanties de connexité
(p.ex. que les entités du graphe soient liées par des relations du vocabulaire, que
le graphe résultant ait une densité raisonnable pour une comparaison à des cas
réels).
```

Evaluation – Types d'erreurs et total des fichiers en erreur (F_{err}) par prompt

	Fichier tronqué	Erreur de syntaxe	Erreur de syntaxe liée à un préfixe	Préfixe non défini ou absent	F_{err}
P1	1	**0**	xsd(10), kos(3)	ldp(1)	15
P2	5	2	foaf(1), kos(2), xsd(2), noria(1)	absent(1)	14
P2_1	**0**	1	xsd(2), org(1), noria(2)	devopsprod(1), absent(1)	8
P3	**0**	1	xsd(4), kos(1), noria-kos(7)	-	13
P3_1	2	2	xsd(6), noria-kos(4)	absent(1)	15
P4	1	2	foaf(2)	ldp(4), absent(2)	11
P4_1	2	2	kos-app(1)	absent(1)	6
P4_1*	2	1	foaf(1), voaf(1), vann(1)	absent(5)	11
P4_1	1	1	xsd(1)	absent(1)	**4**
P4_1*	1	3	voaf(1)	absent(6)	11
P4_1	**0**	2	kos(1), foaf(2), voaf(1)	-	6
P4_1*	1	3	noria(3), xsd(2)	absent(3)	12

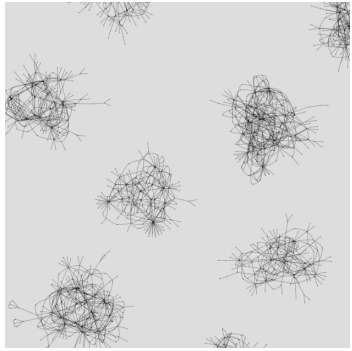
Evaluation – Non-conformités sur les DatatypeProperty

Cette table liste, pour chaque DatatypeProperty incriminée, le nombre de fichiers où le littéral est en erreur (F_{err}), le nombre de fichiers dans lesquels le littéral apparaît (F_{lit}), et le pourcentage F_{err}/F_{lit} correspondant.

DatatypeProperty	F_{err}	F_{lit}	%
`noria:agentInstruction`	14	14	100,0
`noria:logText`	7	7	100,0
`noria:agentWorkingHours`	12	14	85,7
`noria:networkInterfaceDescription`	5	8	62,5
`noria:documentHRef`	6	12	50,0
`noria:networkLinkType`	1	12	8,3
`noria:troubleTicketSupervisionTool`	1	13	7,7

Evaluation – Influence du renommage des nœuds sur le graphe résultant

Figures : (gauche) tous les homonyme renommés ; (droite) aucun homonyme renommé.



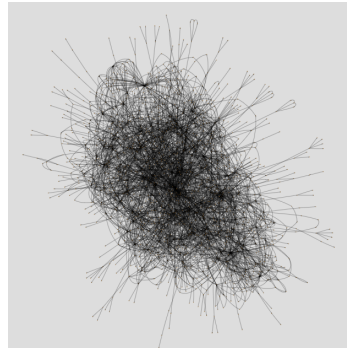


Table : influence du paramètre occ_{max} sur le nombre de nœuds (n) et la densité ($\Delta(\mathcal{G})$) du graphe résultant de la concaténation des 14 fichiers-graphes syntaxiquement corrects générés par le prompt P4_1.

occ_{max}	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
n	1427	1451	1480	1507	1548	1591	1635	1677	1722	1783	1847	1924	2019	2226
$\Delta(\mathcal{G})$	0,0021	0,0020	0,0020	0,0019	0,0018	0,0017	0,0016	0,015	0,0014	0,0013	0,0013	0,0012	0,0010	0,0009