Rankingdom : une architecture coopérative pour l'analyse à la demande de Wikidata

Hassan Abdallah*, Béatrice Markhoff**, Manon Ovide*, Louise Parkin*, Arnaud Soulet*

*Université de Tours, LIFAT, Blois
**Université de Tours, CITERES UMR 7324, Blois
firstname.lastname@univ-tours.fr

Résumé. Nous présentons Rankingdom, une application Web d'analyse des entités de Wikidata destinée aux travailleurs de la connaissance. Son objectif est de positionner chaque entité considérée dans son domaine, et ce en la comparant à d'autres entités similaires en exploitant plusieurs indicateurs inspirés de l'infométrie. Par soucis de frugalité, nous nous appuyons sur une architecture coopérative qui décentralise le calcul à la demande des analyses sur les clients, mais centralise leurs résultats. Pour rendre possible le calcul à la demande des analyses malgré l'accord d'usage non abusif de Wikidata, les clients bénéficient d'un nouveau moteur de parallélisation des requêtes SPARQL, nommé SParaQL. Nous montrons l'efficacité de SParaQL sur un test de performance et de notre architecture coopérative avec le déploiement réel de Rankingdom.

1 Introduction

Les énormes graphes de connaissances du Web des données comme Wikidata (Vrandečić et Krötzsch, 2014) constituent une source de connaissances inestimable déjà largement utilisée pour la recherche d'information et la recommandation (Zou, 2020). Cependant, plusieurs travaux récents soulignent leur potentiel sous-exploité pour les travailleurs de la connaissance (Weikum, 2021; Zhu et al., 2024). En science des réseaux, l'avènement d'une nouvelle forme de réseaux complexes a entrainé systématiquement la proposition d'outils spécifiques de mesures pour l'analyse des entités décrites par ces données (e.g., la bibliométrie pour les réseaux d'articles scientifiques ou la webométrie pour les pages Web) (Thelwall, 2008). Naturellement, en s'inscrivant dans cette direction, les graphes de connaissances apparaissent donc comme une source potentielle prometteuse, d'autant qu'ils couvrent des domaines où il n'existe pas déjà d'outils d'analyses quantitatives. Prenons par exemple la peinture, il serait intéressant d'utiliser les connaissances de Wikidata pour mesurer et analyser son positionnement par rapport aux autres arts, ses principales périodes, ses peintres influents, etc. Malheureusement, à notre connaissance, il n'existe pas d'outil simple pour les travailleurs de la connaissance pour analyser les entités d'un graphe de connaissances.

Construire un outil d'analyse d'un graphe de connaissances aussi large que Wikidata entraîne deux défis majeurs. Premièrement, l'extrême variété des entités soulève la difficile question d'un schéma d'analyse transdisciplinaire. Par exemple, rien que le domaine de la peinture

implique des entités de nature hétérogène : objets (pinceau), événements (exposition), concepts (esthétisme, beauté), etc. Il n'est pas possible de prévoir des schémas d'analyses pour chacun des types en sélectionnant manuellement les relations adéquates pour mesurer l'importance d'une entité. Par exemple, il parait normal de compter le nombre de créations pour comparer deux artistes tandis que le nombre d'artistes peut permettre de comparer deux domaines artistiques. Deuxièmement, le calcul des indicateurs pose des problèmes de frugalité à cause du volume de données. Bien sûr, il est possible de calculer hors ligne les indicateurs pour toutes les entités existantes en utilisant un cadre de calcul distribué dédié aux mégadonnées comme Hadoop ou Spark (Auer et al., 2012; Barbieri et al., 2010; Lehmann et al., 2017). Cette approche répétée régulièrement sur un nouveau dump est particulièrement coûteuse en téléchargement, stockage et temps de calcul. Mais surtout, traiter exhaustivement hors ligne toutes les entités consomme inutilement beaucoup d'énergie pour calculer des informations sur des entités dont la plupart ne seront jamais interrogées en ligne (Soulet, 2023). Par frugalité, il serait judicieux de calculer les indicateurs à la demande en s'appuyant directement sur le point d'accès public SPARQL. Mais, comme ces points d'accès n'utilisent pas la préemption, ils mettent en oeuvre des accords d'usage non abusif (*fair use policy*) qui limitent la fréquence d'exécution et la complexité des requêtes exécutées (Verborgh et al., 2016; Grall et al., 2020).

Dans cet article, nous présentons Rankingdom qui surmonte ces défis pour construire un outil d'analyse à la demande des 110 millions d'entités de Wikidata. Pour contourner l'accord d'usage non abusif de Wikidata, son originalité est de décentraliser côté client le calcul des indicateurs, inspirés de l'infométrie. Ainsi :

— Nous présentons plusieurs indicateurs complémentaires pour situer ou évaluer n'importe quelle entité de Wikidata au sein de son domaine. Leur mise en oeuvre constitue des tâches d'analyses complexes difficiles à traiter avec les méthodes traditionnelles.
— Nous proposons une architecture d'analyse à la demande qui décentralise le calcul des analyses sur les clients mais centralise leurs résultats par frugalité. Cette architecture repose sur un moteur de parallélisation de requêtes SPARQL côté client (SParaQL) qui rend possible le calcul en ligne de l'analyse d'une entité.
— Nous montrons l'efficacité de SParaQL sur un test de performance et nous étudions l'efficacité de notre architecture à travers son déploiement réel, accompagné de la mise en cache de plus de 10 000 entités.

La section 2 présente la méthodologie de notre cadre infométrique et les tâches d'analyse résultantes. Dans la section 3, nous décrivons l'architecture coopérative permettant de les traiter efficacement grâce au moteur SParaQL. Nous évaluons notre approche dans la section 4 avant de conclure dans la section 5.

2 Cadre infométrique pour l'analyse d'entités

2.1 Méthodologie

Pour rappel, un graphe de connaissances sur un ensemble d'arêtes $\mathcal{R}$ (pour les relations) et un ensemble de sommets $\mathcal{E}$ (pour les entités et les valeurs) est un ensemble d'arêtes étiquetées $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ (correspondant aux faits). Nous écrivons les faits sous la forme $\langle s, r, o \rangle \in \mathcal{K}$, où r est la relation, s est le sujet et o est l'objet. La figure 1 décrit l'entité `art of painting` avec ses faits, où elle est sujet (resp. objet) re-

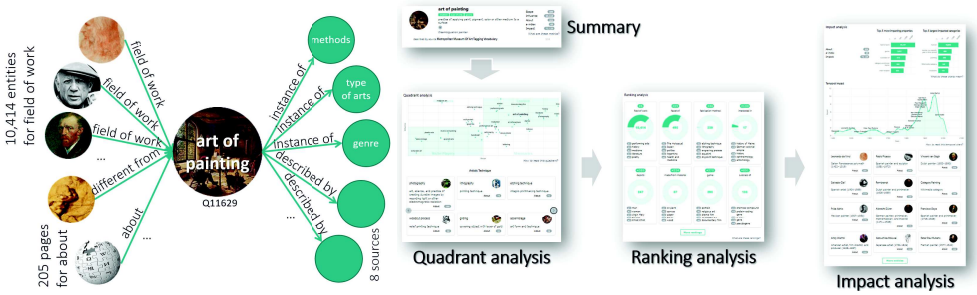

FIG. 1 – À gauche : `art of painting` dans Wikidata, À droite : `art of painting` dans Rankingdom

présentés en liens sortants à droite (resp. liens entrants à gauche). Étant donnée une relation r, $\langle s, r^{-1}, o\rangle \in \mathcal{K}$ signifie que $\langle o, r, s\rangle \in \mathcal{K}$ où r^{-1} est la relation inverse de r. En outre, $r_{\mathcal{K}}(s)$ (ou simplement $r(s)$ si le contexte est clair) est l'ensemble des objets associés au sujet s pour la relation r dans $\mathcal{K}$. Typiquement, `instance of`$($`art of painting`$)$ retourne l'ensemble $\{$`method`, `type of arts`, `genre`$\}$. Il est aussi possible de considérer les 10 414 entités dont le champ de travail est la peinture en considérant la propriété inverse : `field of work`$^{-1}(`art of painting`)$.

Dans ce contexte, Rankingdom vise à présenter à l'utilisateur les mêmes analyses quelle que soit l'entité considérée, et ce malgré leur diversité. En effet, cette structuration commune à toutes les entités, illustrée par la figure 1, facilite la comparaison et l'exploration des entités. Nous avons opté pour une narration de données simple commençant par un bref résumé de l'entité analysée, puis allant d'une analyse globale (la positionnant dans son domaine) jusqu'à une analyse très fine (détaillant ses entités marquantes). Pour chacune des trois analyses détaillées dans la section suivante, nous nous appuyons sur 3 principes : *Mesurer* l'importance des entités en utilisant plusieurs comptages de faits (**P1**) ; *Sélectionner* les informations à proposer à l'utilisateur en s'appuyant sur ces mesures (**P2**) et *Ordonner* les informations en les triant suivant divers ordres et catégories pour faciliter l'interprétation (**P3**). Dans la suite nous rappellerons entre parenthèses le principe (P1, P2 ou P3) guidant le point présenté.

Nous reprenons des techniques de comptage largement éprouvées en infométrie et en recherche d'informations (Thelwall, 2008). En effet, de manière générale, l'importance d'une entité est proportionnelle à sa concentration de faits. Bien que tous les faits d'un graphe de connaissances n'aient pas la même valeur informationnelle et que certains faits non-représentés au sein du graphe pourraient être inférés, il reste qu'avoir de nombreux faits pour une relation indique l'importance de l'entité pour cette relation. Cette sémantique de chaque comptage induite par la relation distingue notre approche des méthodes traditionnelles. Contrairement à l'infométrie traditionnelle, plusieurs milliers de types de liens différents coexistent au sein de Wikidata. En comparaison, il y a seulement quelques types de liens en bibliométrie comme `author`, `publisher` et `cite`. Le tableau 1 définit les principaux indicateurs utilisés au sein de Rankingdom, que nous décrivons et exploitons dans la section suivante.

Indicateur	Formule	Définition
Quadrant analysis		
Scope	$scope(e) = \sum_{r \in \mathcal{R}} \lvert r(e) \rvert$	Nombre de faits ayant pour sujet l'entité e
Influence	$influence(e) = \sum_{r \in \mathcal{R}} \lvert r^{-1}(e) \rvert$	Nombre de faits ayant pour objet l'entité e
Ranking analysis		
Influence de r	$influence(e, r) = \lvert r^{-1}(e) \rvert$	Nombre de faits pour r avec l'entité e comme objet
Impact analysis		
Influence about	$about(e) = influence(e, \texttt{schema:about})$	Nombre de faits pour `schema:about` avec l'entité e comme objet
Impact	$impact(e) = \sum_{\langle s,r,e \rangle \in \mathcal{K}} about(s)$	Somme de l'influence $about$ de chacune des entités impactées (entité sujet pour au moins un fait avec l'entité e comme objet)
a-index	$a\text{-}index(e) = \max\{a \in \mathbb{N} \text{ such that } \lvert\{\langle s,r,e \rangle \in \mathcal{K} : about(s) \geq a\}\rvert \geq a\}$	Nombre maximum a d'entités impactées ayant au moins a comme influence $about$

TAB. 1 – Indicateurs d'analyse utilisés par Rankingdom

2.2 Des mesures aux tâches d'analyse

Quadrant analysis Cette première analyse très générale, point 1 de la figure 1, vise à situer l'entité analysée au sein de son domaine en la comparant à d'autres entités similaires à la manière du diagramme Magic Quadrant de Gartner (Bresciani et Eppler, 2008).

Intuitivement, les relations ont un fonctionnement asymétrique. La plupart des relations sont fonctionnelles (i.e., une entité sujet est associée à une unique entité objet) alors que les relations inverses le sont rarement (i.e., plusieurs entités sujets pointent vers un même objet). De plus, une entité sujet a tendance à pointer vers des entités objets plus générales, plus conceptuelles, plus anciennes dans le graphe, notamment lorsqu'il a été produit collaborativement (Abdallah et al., 2024). Dans ce contexte, il est judicieux de mesurer séparément le nombre de liens entrants et le nombre de liens sortants (cf. figure 1). Ainsi, nous utilisons deux indicateurs très classiques (qui s'appliquent à n'importe quelle entité) à savoir le nombre de faits où l'entité analysée est sujet (scope) et objet (influence) (**P1**). Le scope et l'influence estiment respectivement l'importance d'une entité par essence et son impact au sein du graphe de connaissances. L'analyse par quadrants projette les entités similaires à celle analysée dans un espace bidimensionnel avec le scope en abscisse et l'influence en ordonnée. Plus l'entité est proche du coin supérieur droit, plus elle est considérée comme importante. Comme pour le diagramme Magic Quadrant de Gartner (Bresciani et Eppler, 2008), ces deux dimensions subdivisent l'espace en quatre quadrants.

Pour que la comparaison soit pertinente, les entités projetées dans les quadrants appartiennent au même domaine que l'entité à analyser (**P2**). De manière générale, il n'est pas aisé de déterminer automatiquement le domaine d'une entité. Au premier abord, il semblerait possible de s'appuyer sur une forme de schéma à travers les relations comme `instance of` pour identifier ce domaine. Malheureusement, les classes sont régulièrement trop vastes comme pour la classe humain qui contient plus de 10 millions d'instances. Pour les humains, il serait bien possible d'exploiter d'autres relations comme `occupation` ou `field of work`, mais il n'est pas simple de généraliser une telle démarche pour les dizaines de milliers de classes de Wikidata. Plutôt que de déterminer un domaine grâce à une relation spécifique, nous recherchons les k entités les plus similaires à l'entité analysée. Il existe de nombreuses mesures de similarité dans la littérature pour comparer deux entités (Cheniki et al., 2016). Pour notre part, nous exploitons des indices topologiques. Ainsi, une entité pour être similaire à celle analysée doit être l'objet d'au moins un même sujet et d'une même relation. Ensuite, les entités parta-

Requête de similarité q_{sim}

```
SELECT ?similar (COUNT(*) AS ?nb) WHERE {
 SELECT DISTINCT ?similar ?q ?r WHERE {
  ?s ?p wd:Q11629, ?similar.
  ?similar ?q ?r.
  wd:Q11629 ?q ?r.
 } }
GROUP BY ?similar
ORDER BY DESC(?nb)
```

Décomposition de la requête de similarité $D_{q_{sim}}$

q^d_{sim} (file d'attente principale)

```
SELECT ?similar WHERE {
 ?s ?p wd:Q11629, ?similar.
 FILTER(STR(?similar) != "wd:Q11629")
}
```

q^{Q8242}_{sim} (file d'attente « similarité »)

```
SELECT (COUNT(*) AS ?nb) WHERE {
 wd:Q8242 ?q ?r.
 wd:Q11629 ?q ?r.
}
```

...

FIG. 2 – Requête de similarité et sa décomposition

geant en tant que sujet le plus de faits (i.e., même relation et même objet) avec l'entité analysée sont considérées comme les plus similaires. La requête SPARQL q_{sim} reportée dans la figure 2 permet de calculer les entités similaires à la peinture (`Q11629`) ce qui retourne : peinture, musique, mythologie, littérature, etc. L'exécution naive de cette requête à partir du point d'accès SPARQL de Wikidata échoue du fait de son accord d'usage non abusif, nous y revenons en section 3.2. Les différentes entités présentées dans l'analyse par quadrant sont également regroupées en catégories plus fines au sein d'un carrousel (**P3**).

Ranking analysis Cette deuxième analyse , point 2 de la figure 1, vise à faire émerger les domaines d'influence de l'entité et à la positionner au sein de ces domaines. Plutôt que de considérer l'ensemble des liens entrants, il est pertinent de séparer l'influence de l'entité analysée pour chacune des relations afin d'identifier ses forces. De manière intéressante, la relation sur laquelle porte le comptage induit une sémantique particulière (**P1**). Par exemple, $\mathit{influence}(\texttt{art of painting}, \texttt{field of work}) = 10414$ quantifie le nombre de peintres. Bien sûr, toutes les relations ne sont pas forcément pertinentes pour classer une entité. Par exemple, la relation `different from` n'est pas intéressante pour mesurer l'influence de la peinture. De plus, étant donnée l'hétérogénéité des entités, il n'est pas possible de prévoir à l'avance quelles relations seront pertinentes pour chaque entité (ou même type d'entités).

En pratique, une relation sera d'autant plus intéressante pour identifier un domaine d'influence d'une entité si cette dernière dispose de nombreux faits pour cette relation. Mais, une entité peut avoir de nombreux faits pour une relation pourtant inintéressante. Par exemple, de nombreuses entités "Long Lake" existent avec plusieurs centaines de liens `different from` entre eux. Il est clair que ce nombre de liens ne traduit pas une influence directe de l'entité. Pour cette raison, en plus d'un seuil minimum de faits m, nous nous appuyons sur la modélisation d'une relation proposée par Abdallah et al. (2024). Selon ce modèle, la distribution du nombre de faits pour une relation r suit une loi de puissance d'exposant $1 + 1/(1 - n_o/n_r)$ où $n_o = |\{o \in \mathcal{E} \ : \ (\exists s \in \mathcal{E})(\langle s, r, o\rangle \in \mathcal{K})\}|$ et $n_r = |\{\langle s, r, o\rangle \in \mathcal{K} \ : \ s \in \mathcal{E} \wedge o \in \mathcal{E}\}|$. Cet article montre que dans le cadre d'une loi de puissance où $n_o/n_r \leq 0.5$ la variance n'est pas définie, ce qui implique que la distribution des faits de cette relation est discriminante, ce qui rend cette relation intéressante pour l'analyse des entités qu'elle caractérise.

Ainsi, pour une entité e, Rankingdom retient dans la partie *Ranking analysis* toutes les relations qui vérifient $n_o/n_r \leq 0.5$ et pour lesquelles e a au moins $m = 10$ faits (**P2**). Par exemple avec ces critères, 10 relations (dont `field of work`) sont retenues pour la pein-

ture. Pour chacune des relations retenues, il est alors précisé le nombre de faits de l'entité par rapport au maximum possible dans Wikidata. Le rang de l'entité est aussi calculé en listant les 5 premières entités du classement. Par exemple, la peinture est le sixième plus important `field of work` après `performing arts`, `history`, `medecine`, `literature` et `poetry`. Les classements issus de chaque relation sont énumérés suivant la position de l'entité analysée afin de mettre en avant les domaines d'influence les plus forts (**P3**).

Impact analysis Cette troisième analyse, point 3 de la figure 1, met en avant les entités les plus marquantes participant à l'influence de l'entité analysée. Pour cela, nous recourons à la relation `schema:about` qui contient plus de 90 millions de faits au sein de Wikidata et couvre les entités de tous les domaines. En effet, cette relation référence tous les projets de la fondation Wikimedia (i.e., pages Wikipedia, Wikiquote, etc) concernant une entité ce qui permet de mesurer l'importance de l'entité dans toutes les langues. Par exemple, pour la peinture, l'entité influencée la plus marquante est Léonard de Vinci avec $about(\texttt{Da Vinci}) = 291$. Rankingdom s'appuie sur le score $about$ pour détailler finement l'influence de l'entité analysée en reposant sur deux mesures (**P1**) : le a-index et l'impact.

Premièrement, Rankingdom calcule le a-index de l'entité analysée qui correspond au nombre maximum a d'entités influencées avec au moins a pour leur score $about$. Comme pour le h-index en bibliométrie (Thelwall, 2008), cet indicateur vise à capturer avec un seul nombre la quantité et la qualité de l'influence. Le a-index de la peinture est de 69 car elle dispose de 69 entités influencées avec au moins un score $about$ de 69 (comme de Vinci). Ces entités les plus marquantes (car participant au a-index de l'entité (**P2**)) sont énumérées à la fin de cette analyse par score $about$ décroissant (**P3**). Deuxièmement, Rankingdom détaille l'impact de l'entité analysée suivant la relation, le type et le temps (**P3**). L'impact est juste la somme des score $about$ de l'ensemble des entités influencées (e.g., 72 129 pour la peinture). Plusieurs tâches d'analyse raffinent cette répartition. Pour la peinture, on observe ainsi que la relation la plus marquante est `field of work` et le type le plus marquant est `human`. Enfin, un diagramme d'impact temporel montre la répartition de l'impact au cours du temps en s'appuyant sur l'ensemble des relations dont l'objet est une date. Par exemple, pour la peinture, il permet facilement à un utilisateur d'observer les périodes de la renaissance italienne, de la peinture flamande, du mouvement impressionniste, etc. Les principaux peintres participant au a-index de la peinture sont aussi reportés (comme Léonard de Vinci ou Frida Khalo).

2.3 Défi de l'accord d'usage non abusif pour les tâches d'analyse

Les tâches d'analyse de Rankingdom comme la génération du quadrant ou le diagramme d'impact temporel requièrent l'exécution de nombreuses requêtes SPARQL complexes impliquant des sous-requêtes et des calculs d'agrégats avec regroupements. Malheureusement, ces requêtes ne peuvent pas être exécutées directement sur le point d'accès public SPARQL de Wikidata qui est volontairement bridé pour en faciliter le partage. En effet, un accord d'usage non abusif $\mathcal{P}$ est un ensemble de contraintes que doit respecter l'exécution d'un ensemble de requêtes Q. Par exemple, l'accord d'usage non abusif du point d'accès public SPARQL de Wikidata $\mathcal{P}_W$ empêche l'exécution des requêtes trop coûteuses et limite le nombre de requêtes sur une fenêtre temporelle glissante. On note c le seuil maximal de coût par requête, d la durée de la fenêtre glissante et l le nombre maximal de requêtes sur cette durée. Nous décrivons dans la

section suivante les choix architecturaux originaux absolument nécessaires pour relever le défi de l'exécution des requêtes complexes respectant l'accord d'usage non abusif de Wikidata.

3 Architecture coopérative fondée sur le moteur SParaQL

3.1 Aperçu de l'approche

Notre architecture suit un schéma classique d'interrogation. Le serveur Rankingdom identifie l'entité recherchée par le client en s'appuyant sur le moteur de recherche de Wikipedia. Si l'entité identifiée a été analysée récemment et est donc stockée dans le cache, le résultat de l'analyse est retourné au client. Le client pourra alors directement afficher cette analyse en exploitant également des ressources disponibles sous Wikipedia comme des images.

Cependant, il y a deux aspects originaux dans cette architecture. Premièrement, par souci de frugalité, nous utilisons un cache plutôt qu'une base de données contenant l'intégralité des analyses précalculées pour chacune des entités. Cette approche évite d'avoir à traiter en amont des énormes volumes de données pour produire des analyses correspondant à des millions d'entités. En effet, la grande majorité de ces entités est peu populaire et ne sera jamais interrogée induisant un coût de traitement et de stockage inutiles (Soulet, 2023). Avec cette approche, si l'entité recherchée n'a pas été précédemment interrogée, nous devons calculer en ligne son analyse ce qui a l'avantage de s'appuyer sur les données fraîches issues du point d'accès public de Wikidata continuellement enrichi. Deuxièmement, le calcul en ligne de l'analyse se fait côté client et non côté serveur pour pouvoir passer à l'échelle. Il peut paraître contre-intuitif de s'appuyer sur une machine moins puissante pour effectuer des calculs. Néanmoins, cette décentralisation se justifie par l'accord d'usage non abusif des points d'accès publics qui limitent le nombre d'interrogations par client SPARQL. Faisons l'hypothèse que 100 analyses doivent être calculées en même temps, le serveur Rankingdom serait limité par l'accord d'usage non abusif $\mathcal{P}_W$ à l requêtes sur l'intervalle de temps d (voir la section 2.3). Pour ce même accord d'usage non abusif, 100 clients peuvent simultanément interroger $100 \times l$ requêtes sur cette même période d. A la fin, il est tout de même important que le client retourne le résultat de ses analyses au serveur pour alimenter le cache. Cette coopération entre les clients évite des calculs redondants et accélère le temps de réponse pour les entités fréquemment recherchées.

La suite de cette section présente le fonctionnement de SParaQL qui est un moteur de parallélisation de requêtes SPARQL côté client. Il optimise côté client l'exécution des requêtes pour satisfaire les accords d'usage non abusif des points d'accès public SPARQL. Formellement, pour une tâche d'analyse nécessitant l'ensemble de requêtes Q, notre approche vise à exécuter un ensemble de requêtes Q' minimisant le temps d'exécution et permettant de calculer tous les résultats issus de Q. A notre connaissance, il s'agit du premier moteur qui optimise la parallélisation de requêtes côté client. En effet, les approches traditionnelles optimisent l'exécution des requêtes SPARQL en modifiant le fonctionnement du serveur – avec parfois un client intelligent (Verborgh et al., 2016; Grall et al., 2020). En particulier, une difficulté inhérente est de ne pas pouvoir estimer la fonction de coût C pour les requêtes (connue seulement côté serveur SPARQL). Pour cette raison, nous travaillons avec une approximation de l'accord d'usage non abusif en ayant une fonction de coût pessimiste $\widetilde{C}$ de sorte que $\widetilde{C}(q) \geq C(q)$ pour toute requête q. Par simplicité, nous conservons toutefois la notation C dans la suite.

3.2 Décomposition en requêtes à faible coût

Pour une tâche d'analyse impliquant les requêtes Q, il est impératif d'éviter qu'une requête $q \in Q$ ne remette en question l'ensemble de la tâche en étant rejetée par l'accord d'usage non abusif. Typiquement, pour Wikidata, toutes les requêtes doivent avoir un coût C inférieur à c. Pour cette raison, il est préférable de décomposer les requêtes de Q, qui risquent d'être trop coûteuses, en plusieurs requêtes de coût plus faible en s'inspirant d'approches mise en oeuvres habituellement côté serveur (Yan et Larson, 1995). Plus précisément, l'ensemble de requêtes Q' est une décomposition de Q si le résultat de chaque requête de Q peut être calculé à partir de ceux de Q', i.e. pour tout $q \in Q$, il existe une fonction d'agrégat agg telle que $q \equiv agg(Q')$. Ainsi, les requêtes D_q décomposant q seront moins coûteuses (i.e., $C(q') \leq C(q)$ pour $q' \in D_q$) favorisant les chances de satisfaire l'accord d'usage non abusif $\mathcal{P}$.

Illustrons ce principe de décomposition sur notre requête q_{sim} reportée à la figure 2. Elle peut se décomposer avec une requête énumérant les entités similaires q^d_{sim} et une requête comptant le nombre de faits en commun pour chaque entité similaire (par exemple, $q^{\texttt{Q8242}}_{sim}$ pour l'entité `Q8242`). Plus précisément, la première étape avec q^d_{sim} énumère les entités similaires sans utiliser un opérateur bloquant (comme `DISTINCT`). Ensuite, la seconde étape prend chaque entité du résultat précédent (comme par exemple la littérature avec `Q8242`) pour compter le nombre de faits sortants en commun avec l'art de la peinture avec $q^{\texttt{Q8242}}_{sim}$. Enfin, le calcul du résultat final de la requête q_{sim} peut être reconstitué en faisant l'union des résultats et en les triant (i.e., $\bigcup q^{\texttt{Q}_i}_{sim}$). Malheureusement, cette décomposition $D_{q_{sim}} = \{q^d_{sim}, q^{\texttt{Q8242}}_{sim}, \dots\}$ s'expose à deux écueils majeurs. 1) Il faut éviter qu'une entité similaire qui apparait plusieurs fois dans le résultat de la première requête entraine des requêtes de comptage redondantes. 2) Il risque d'y avoir une profusion de requêtes dans la seconde étape (e.g., plus de 2000 pour `art of painting`). L'exécution une à une de ces requêtes prendrait un temps considérable.

3.3 Parallélisation et union

L'approche par décomposition conduit à une énorme quantité de requêtes à exécuter. Naturellement, il est possible de paralléliser leur exécution. Néanmoins, l'accord d'usage non abusif de Wikidata $\mathcal{P}_W$ contraint le nombre d'exécutions par seconde pour un client donné. Pour cette raison, SParaQL réduit le nombre de requêtes à exécuter en exploitant à la fois un cache local et l'union des requêtes modestes.

Cache local Le cache local est un cache côté client pour éviter d'exécuter deux fois la même requête. Typiquement, la première exécution de la requête de comptage $q^{\texttt{Q8242}}_{sim}$ sera exécutée normalement tandis que leur potentielles exécutions ultérieures retourneront directement le résultat conservé en cache. Ici, il serait possible d'empêcher manuellement des exécutions multiples de la même requête en retenant celles déjà exécutées dans une liste. En revanche, l'automatisation du cache local est essentielle pour utiliser les frameworks réactifs de développement Web. Ces derniers reposent sur des composants isolés dont la réactivité répète régulièrement les mêmes requêtes non synchronisables.

Union Pour diminuer drastiquement le nombre de requêtes à exécuter, nous groupons les requêtes entre elles en utilisant l'opérateur `UNION`. Il est ainsi possible pour des requêtes très simples comme la requête de comptage $q^{\texttt{Q8242}}_{sim}$ d'en grouper plusieurs ensemble. Bien sûr, le coût de la requête groupée doit rester raisonnable pour satisfaire l'accord d'usage non abusif.

Méthode	Nbr. d'entités analysées	Tps. total moy.	Tps. de réponse moy.	Nbr. moy. de req.	Tps. moy. par req.
Classic	224 (22%)	4,84	4,84	1	4,84
NoUnion	999 (98%)	174,64	2,77	964	0,18
SParaQL	1 002 (98%)	62,76	3,05	218	0,29

TAB. 2 – Evaluation des performances sur la requête q_{sim}

Par ailleurs, le temps d'exécution des requêtes groupées est plus grand que celui des requêtes simples qui les composent ce qui s'avère intéressant. En effet, avec des temps d'exécution trop courts, la limite de fréquence force à attendre entre deux requêtes ce qui fait perdre du temps d'exécution côté serveur. En revanche, des requêtes avec un temps d'exécution long permettent de profiter pleinement du temps d'exécution disponible côté serveur.

Files d'attente et équilibrage de la charge des requêtes Pour préserver l'interactivité, il est important d'éviter des requêtes ou des tâches d'analyse bloquantes. En effet, au sein de la même page, plusieurs tâches d'analyse distinctes doivent être menées en parallèle (par exemple, le calcul du quadrant et le diagramme d'impact temporel). Pour cela, certaines requêtes importantes nécessitent d'être exécutées rapidement tandis que d'autres potentiellement moins urgentes doivent être exécutées au compte-goutte. Ainsi, SParaQL dispose d'un système de files Q_i pour multiplexer plusieurs tâches d'analyse. Le mécanisme de groupement de requêtes décrit ci-dessus opère à l'intérieur de chacune des files tandis que le cache local est partagé. Une file peut être utilisée par plusieurs tâches (e.g., la file principale contient des requêtes pour lancer plusieurs tâches d'analyse dont le calcul de la similarité) et une tâche peut exploiter plusieurs files (e.g., le calcul de la similarité s'appuie sur la file principale et une file dédiée). Par défaut, le système d'équilibrage exécute successivement la première requête SPARQL (après `UNION`) en attente dans chacune des files, mais il est aussi possible d'ajouter des priorités. En distribuant les tâches d'analyse sur plusieurs files, le temps de réponse d'affichage du premier résultat pour chacune de ces tâches est ainsi relativement rapide.

4 Étude expérimentale

Cette section évalue l'efficacité de SParaQL en s'appuyant sur le traitement de la requête de similarité (en section 4.1) et sur le déploiement de Rankingdom (en section 4.2). Il est à noter que les temps d'exécution sont indépendants de la configuration de la machine côté client car les traitements bloquants sont effectués sur le point d'accès public SPARQL de Wikidata. Nous autorisons 5 interrogations SPARQL en parallèle. Côté client, nous prenons comme modèle de coût $C(q) = 1$ si la requête q ne comporte ni sous-requête, ni clause complexe (e.g., `GROUP BY` ou `DISTINCT`) et $C(q) = 5$ sinon. Par ailleurs, nous fixons les seuils suivants : coût maximal $c = 5$ et délai inter-requêtes $d/l = 150$ ms. Le code source est disponible sur le dépôt Git : `https://scm.univ-tours.fr/soulet/rankingdom`.

4.1 Test de performance sur la requête de similarité

Cette évaluation porte sur le traitement de la requête de similarité pour les 1 017 entités des articles de Wikipedia devant être présents dans toutes les langues. Ces entités correspondent

aux entités nommées les plus significatives et aux concepts universels. Pour cette évaluation, nous comparons notre méthode (utilisant la décomposition $D_{q_{sim}}$) à deux baselines : notre méthode sans l'union (NoUnion) et une exécution de la requête non décomposée q_{sim} (Classic). Nous n'avons pas comparé notre approche avec les clients intelligents (Verborgh et al., 2016; Grall et al., 2020) qui ne traitent pas les sous-requêtes. Pour les 3 méthodes, le tableau 2 reporte le nombre d'analyses réussies (avec le pourcentage), le temps total moyen, le temps de réponse moyen, le nombre moyen de requêtes et le temps moyen d'exécution d'une requête.

Temps d'exécution La méthode SParaQL est moins rapide que Classic qui ne fonctionne que pour les entités portant sur peu de données et qui sont donc les plus faciles à traiter. SParaQL met en moyenne 62,7 secondes contre 4,8 secondes pour Classic, mais cette dernière avec un taux de succès de 22% n'est pas viable. En effet, pour chaque entité, la méthode Classic ne nécessite qu'une seule requête qui est souvent rejetée car bien trop coûteuse. La méthode SParaQL est environ 3 fois plus rapide par rapport à la méthode NoUnion. Cela s'explique à la fois par un moins grand nombre de requêtes à exécuter et par un multiplexage des requêtes plus efficace dû aux requêtes longues (0,29 seconde contre 0,18 pour NoUnion).

Temps de réponse Pour les applications interactives, le temps de réponse est souvent plus pertinent que le temps d'exécution total car l'utilisateur apprécie avoir un retour rapide et souvent, les premiers résultats approximent assez bien le résultat exact obtenu à la fin de l'exécution. Le temps de réponse des méthodes SParaQL et NoUnion sont environ de 3 secondes ce qui est raisonnable pour assurer l'interactivité. Sans surprise, il y a un gain par rapport à la méthode Classic (1,6 fois plus lente) concernant pourtant uniquement les entités les plus faciles à traiter.

4.2 Déploiement réel de l'application Rankingdom

Cette évaluation porte directement sur le déploiement de l'application Web Rankingdom en évaluant les coûts de calcul de l'analyse complète d'une entité et de stockage dans le cache centralisé. Pour cela, nous avons utilisé la phase de déploiement initial où 10 562 entités populaires ont été mises en cache via l'exécution d'un client web interrogeant Rankingdom. Les entités sélectionnées correspondent aux entités de Wikidata appartenant à au moins une des listes suivantes : Exciting women in history (`Q85319493`), Vital articles/Level/4 (`Q6173448`), List of articles all languages should have (`Q5460604`) et les occupations les plus instanciées. A noter que Rankingdom utilise les mêmes identifiants que Wikidata. Par exemple, `https://rankingdom.org/entity/Q5588` se réfère à l'entité Frida Khalo (`wd:Q5588`). Il est aussi possible d'éviter l'utilisation du cache en ajoutant `cache=no` à la query string.

Temps d'exécution L'histogramme de la figure 3 trace la distribution du nombre d'entités traitées entre 10^i et 10^{i+1} requêtes (magenta), secondes (vert) et kilooctets (bleu). En vert, il montre une grande disparité entre les entités même si la majorité des entités nécessitent moins de 100 secondes. En moyenne, il faut 129 secondes pour générer l'intégralité des différentes tâches d'analyses correspondant à ce qui sera stocké dans le cache centralisé pour les futures interrogations. De manière intéressante, dès les premières secondes, les premiers résultats des analyses sont affichés et ils sont ensuite mis à jour. Sur la figure 3 (b), nous observons que globalement, le temps d'exécution est proportionnel au nombre de requêtes à exécuter correspondant à la limite de 5 requêtes par seconde imposée par l'accord d'usage non abusif du point d'accès de Wikidata. Il y a quelques entités dont le temps d'exécution est exceptionnellement élevé. Cela peut être dû à une surcharge du point d'accès entraînant temporairement des

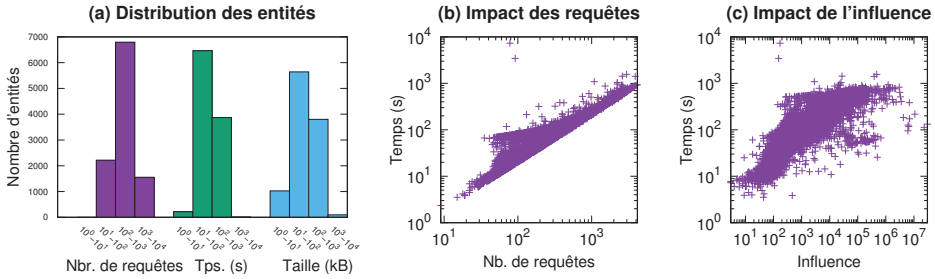

FIG. 3 – Distribution des entités et impact du nombre de requêtes et de l'influence

réponses HTTP erronées obligeant SParaQL à différer l'envoi de ses requêtes. Enfin, sans surprise, plus une entité est jugée importante selon nos indicateurs, plus son temps de traitement est élevé (voir la figure 3 (c) pour l'influence). Car les valeurs des indicateurs augmentent avec le nombre de faits, c'est-à-dire plus de données à traiter et donc plus de requêtes à effectuer.

Taille du cache La figure 3 (a) montre, en bleu, aussi une forte disparité concernant la taille des données associées à chaque entité allant de 928 octets à 2,12 Mo. Au total, l'ensemble des données JSON représente 1 264 Mo. Avant la transmission au serveur, les URI sont écourtées avec le recours à des préfixes (par exemple, `wd:` remplace `http://www.wikidata.org/entity/`) conduisant à une réduction d'environ 30%, soit 860 Mo. Finalement, ces données stockées sous la forme de documents au sein de MongoDB occupent 304 Mo auxquels il faut ajouter 420 ko pour le stockage de l'index. De manière intéressante, ce cache a une taille bien raisonnable. Mais il serait très coûteux de le généraliser aux 113,5M entités de Wikidata dont les 10 562 entités sélectionnées ne représentent que 0,01%.

5 Conclusion

Nous proposons une application permettant de situer l'importance des entités dans leur domaine. Rankingdom propose trois analyses communes à toutes les entités, où se dégagent leurs singularités. Pour calculer les indicateurs nécessaires à ces analyses, malgré l'accord d'usage non abusif de Wikidata qui limite les interrogations du point d'accès, nous distribuons l'interrogation au niveau des clients avec SParaQL. Plus précisément, il s'agit de trouver un équilibre dans le découpage des requêtes pour diminuer le coût de chacune, en conservant des temps d'exécution longs qui permettent la parallélisation en respectant la limite de fréquence d'envoi des requêtes. Cette architecture rend possible l'analyse de plus de 110M d'entités dans une base de connaissances avec environ 16.3B de faits. Dans nos travaux futurs, nous voulons développer de nouveaux indicateurs de classement, pouvant dépendre de l'entité analysée, qui soient plus variés tout en restant explicables. Pour affiner ces indicateurs, nous souhaitons mettre en place une étude d'évaluation de l'outil par les utilisateurs. Pour l'instant, l'architecture coopérative proposée ne gère pas la péremption du cache, mais nous envisageons de compter le nombre de faits entrant ajoutés à une entité et d'invalider le cache lorsque ce nombre est supérieur à un seuil donné. Enfin, le moteur SParaQL présente l'avantage de fonctionner sur les

points d'accès publiques existants. Ainsi, il pourrait être réemployé pour d'autres problèmes nécessitant un grand nombre de requêtes statistiques.

Références

Abdallah, H., B. Markhoff, et A. Soulet (2024). A complex network model for knowledge graphs' relationships.

Auer, S., J. Demter, M. Martin, et J. Lehmann (2012). LODStats – An extensible framework for high-performance dataset analytics. In *EKAW*. Springer.

Barbieri, D. F., D. Braga, S. Ceri, E. D. Valle, et M. Grossniklaus (2010). Querying RDF streams with c-SPARQL. *ACM SIGMOD Record 39*(1).

Bresciani, S. et M. J. Eppler (2008). Gartner's magic quadrant and hype cycle. *Institute of Marketing and Communication Management (IMCA) 2*, 19.

Cheniki, N., A. Belkhir, Y. Sam, et N. Messai (2016). LODS : A linked open data based similarity measure. In *WETICE*, pp. 229–234.

Grall, A., T. Minier, H. Skaf-Molli, et P. Molli (2020). Processing SPARQL aggregate queries with web preemption. In *ESWC*, pp. 235–251. Springer.

Lehmann, J., G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. Ngonga Ngomo, et al. (2017). Distributed semantic analytics using the sansa stack. In *ISWC*, pp. 147–155.

Soulet, A. (2023). Should we consider on-demand analysis in scale-free networks ? In *International Symposium on Intelligent Data Analysis*, pp. 406–418. Springer.

Thelwall, M. (2008). Bibliometrics to webometrics. *Journal of information science 34*(4), 605–621.

Verborgh, R., M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, et P. Colpaert (2016). Triple pattern fragments : a low-cost knowledge graph interface for the web. *Journal of Web Semantics 37*, 184–206.

Vrandečić, D. et M. Krötzsch (2014). Wikidata : a free collaborative knowledgebase. *Communications of the ACM 57*(10), 78–85.

Weikum, G. (2021). Knowledge graphs 2021 : A data odyssey. *VLDB 14*(12), 3233–3238.

Yan, W. P. et P.-B. Larson (1995). Eager aggregation and lazy aggregation. In *VLDB*, pp. 13.

Zhu, Y., X. Wang, J. Chen, S. Qiao, Y. Ou, Y. Yao, S. Deng, H. Chen, et N. Zhang (2024). LLMs for knowledge graph construction and reasoning : Recent capabilities and future opportunities. *WWW 27*(5).

Zou, X. (2020). A survey on application of knowledge graph. In *Journal of Physics : Conference Series*, Volume 1487, pp. 012016. IOP Publishing.

Summary

Rankingdom is a Wikidata entity analysis tool for knowledge workers. Its goal is to position it within its domain by comparing it with other similar entities using several indicators. For this purpose, we propose a cooperative on-demand analysis architecture that decentralizes the computation of analyses, but centralizes their results for frugality. This architecture is based on a client-side SPARQL parallelization engine that deals with the fair use policy. We show the effectiveness of our approach with a benchmark and the deployment of Rankingdom.