WiCleanData: Guaranteeing the Type Consistency of Wikidata by Taxonomy Refinement and Constraint Enforcement

Yiwen Peng
Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
yiwen.peng@telecom-paris.fr

Marc Jeanmougin
Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
marc.jeanmougin@telecom-paris.fr

Thomas Bonald
Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
thomas.bonald@telecom-paris.fr

Abstract

Because of its collaborative nature, Wikidata suffers from errors, inconsistencies, and excessive complexity, such as redundant classes, ambiguity between instances and classes, wrong taxonomic paths, and type constraint violations. The manual curation of these issues is infeasible at scale. To address these challenges, we introduce *WiCleanData*, a refined version of Wikidata with a consistent taxonomy and free from type constraint violations. Specifically, we have designed an automated pipeline that first cleans the taxonomy with language model assistance, then simplifies type constraints by hierarchical aggregation, and finally filters facts accordingly. The resulting knowledge graph, free from any type violation, is made publicly available via a Web interface, enabling easy exploration and downstream applications.

CCS Concepts

• **Information systems** → *World Wide Web*.

Keywords

Knowledge Graphs; Ontology; Taxonomy; Type Constraints; Wikidata; Large Language Models; Data Quality

ACM Reference Format:
Yiwen Peng, Marc Jeanmougin, and Thomas Bonald. 2026. WiCleanData: Guaranteeing the Type Consistency of Wikidata by Taxonomy Refinement and Constraint Enforcement. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM '26), November 07–11, 2026, Rome, Italy*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3799682.3840187

1 Introduction

Wikidata is the largest and most widely used general-purpose Knowledge Graph (KG). Maintained by a large community of contributors, it serves as a core resource for a wide range of applications, including entity linking [24], question answering [6], and information retrieval [30]. However, due to its collaborative nature, Wikidata suffers from errors, inconsistencies and excessive complexity. For instance, as per Wikidata on May 9, 2026, the class *film* is the subclass of many redundant classes, including *visual work*, *audiovisual work*, *video and/or audio work*, *video work*, and has *condition* as one of its top-level classes (see Figure 1a). The inherent complexity of the taxonomy affects property type constraints, which are difficult for contributors to verify and remain often incomplete. For instance, the constrained subject types of the property *notable work* include: (I) mixed class orders (e.g., the metaclass *group of humans*), (III) redundant types (e.g., *imaginary character*, a subclass of *character*), and (II) orphan types with no instantiated facts of the corresponding property (e.g., *artificially intelligent entity*). As a result, there are currently about 6M type-constraint violations in Wikidata, which is one of the major challenges for the community [10]. Many of these facts are actually valid, due to the incompleteness of the constraints. For instance, the fact *Max, notable work, The Mask*, whose subject is an *individual animal*, is incorrectly flagged as a type constraint violation (see Figure 1a).

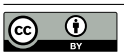
Addressing these issues is challenging due to the scale and the dynamic nature of Wikidata. Its truthy version exceeds 985GB and contains billions of facts that are continuously updated by a large community of editors, tending to further increase the complexity of the taxonomy. In this paper, we present WiCleanData, a refined version of Wikidata with a consistent taxonomy and free from type constraint violations. Specifically, we have designed an automated pipeline that first cleans the taxonomy of Wikidata using Large Language Models (LLMs), then simplifies and refines the type constraints through hierarchical aggregation on the cleaned taxonomy. Finally, the facts are filtered according to the new taxonomy and type constraints. The quality of the resulting knowledge graph is assessed in terms of complexity, conciseness, robustness, coverage, and usefulness. WiCleanData is publicly available via a Web interface[1] for easy exploration and access.

The paper is structured as follows. Section 2 reviews related work, Section 3 describes the automated pipeline used to get WiCleanData from Wikidata, Section 4 presents the evaluation, and Section 5 concludes the paper.

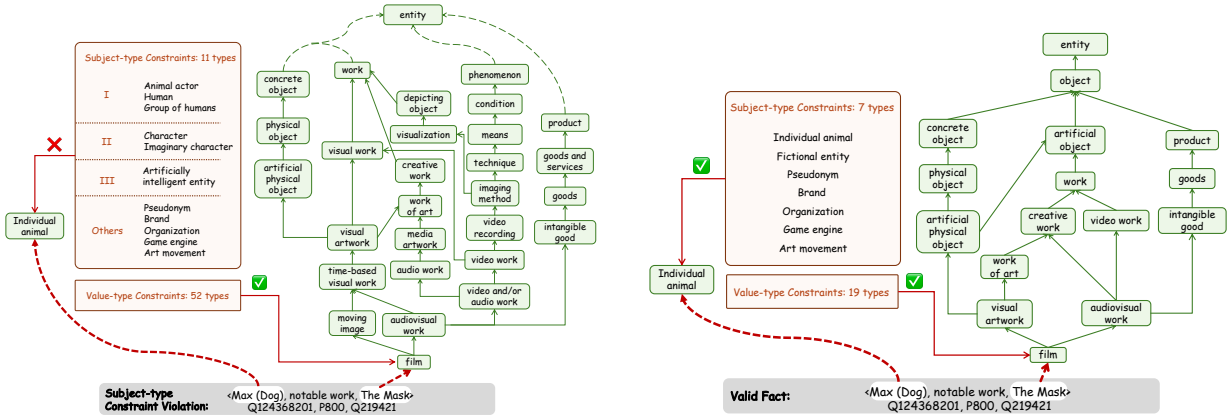
2 Related Work

Knowledge Graph Refinement. KG refinement aims at improving the quality of KGs by detecting and correcting errors [21]. Most existing works focus on the detection of errors [13, 15, 17, 19, 29], while some methods address both detection and repair [22, 34]. For instance, [14] uses crowdsourcing to verify Linked Data quality issues, while [2, 4] requires user interaction to clean violations. KGClean [11] uses active learning to identify incorrect triples and repair them with candidates of maximum propagation-power scores. More recently, [8] feeds KGs directly into LLMs to detect erroneous triples and suggest repairs. However, these methods are too

[1] https://wicleandata.r2.enst.fr/



This work is licensed under a Creative Commons Attribution 4.0 International License.
CIKM '26, Rome, Italy
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2539-5/2026/11
https://doi.org/10.1145/3799682.3840187



(a) An excerpt of Wikidata **(b) WiCleanData**

Figure 1: Repairing type constraint violations of Wikidata by taxonomy cleaning.

resource-intensive to scale to very large KGs like Wikidata. Instead, [33] generates error hyperlinks in Wikipedia by the LinkRank algorithm and then trains an SVM classifier for links correction, while CoCKG [18] exploits disambiguation links and string matching to fix incorrect triples caused by entity confusion. Specifically for Wikidata, [22] mines correction rules from crowdsourced historical edits and uses these rules to fix constraint violations. However, the above methods mainly focus on ABox (instances and facts) and overlook issues in TBox (taxonomies and constraints themselves). [34] studies taxonomy pruning in noisy KGs, but still ignores the type constraints. In this paper, we address this limitation by automatically refining the taxonomy and enforcing proper type constraints.

Efforts on Wikidata. Wikidata [32] is a collaborative KG actively maintained by a large community, but still suffering from taxonomy complexity, redundancies, and inconsistencies [5, 26]. To address these issues, Wikidata manages its data quality through a property constraint model[2], which detects inconsistencies by enforcing or prohibiting specific data patterns [22]. Recently, [10] formally defines Wikidata's repair patterns for each property constraint based on its historical edits. For example, as for the TBox part, the repair primarily involves constraint deletion or class hierarchy additions, especially solving violations of *Value-Type* and *Type* constraints. Since 2023, Wikidata has launched the WikiProject Ontology Cleaning Task Force [3], which documents problem classes, and coordinates contributors to fix high-impact ontology issues. However, Wikidata relies on contributor collaborations, making ontology cleanup a lengthy and incremental process requiring individual commitment and consensus. Other resources have also proposed cleaning Wikidata. For example, YAGO 4.5 [27] attempts to clean Wikidata by manually mapping the upper taxonomy to Schema.org and defining the disjointness classes. DBpedia [3] is derived from Wikipedia (and more recently, Wikidata) with a manually curated ontology. Its automatic extraction process is also prone to inconsistencies [1, 9]. More recently, WiKC [23] proposes a cleaned version of Wikidata taxonomy but ignores constraints and fact-level cleaning. Despite these efforts, Wikidata refinement remains largely manual and time-consuming. In this paper, we introduce a cleaned version of Wikidata by automatically cleaning its taxonomy and constraints, and filtering facts accordingly.

3 Automated Pipeline

We use the truthy version of Wikidata, which contains the best non-deprecated rank for each property, as of May 9, 2026. We exclude the class *scholarly article* that is controversial [27] and contributes to more than one billion facts. Specifically, we remove all instances, facts, and transitive subclasses of *scholarly article*. We also remove all entities without English labels (the vast majority are of type *star*, *galaxy* and *Wikimedia category*), as well as all facts associated with an external identifier[4]. We get around 50M entities and 450M facts.

3.1 Taxonomy Refinement

There is no clear specification for the taxonomy of Wikidata. In principle, it is defined by the *subclassOf* (P279) property. However, this property is often confused with the *instanceOf* (P31) property by Wikidata contributors. To extract the taxonomy, we treat an entity as an actual class if the entity either has at least one subclass or at least one instance with both a label and description; we then build the corresponding taxonomy using the *subclassOf* property. Next, we filter out classes that have no cumulative instances (i.e., classes where neither the class itself nor any of its transitive subclasses have instances). To reduce complexity, we also remove all metaclasses, i.e., instances of *metaclass* or any of its transitive subclasses, except for the important classes *profession*, *occupation*, *taxon*, and *protein*, which we link to their closest parent classes in the taxonomy. Finally, we exclude all instances of *BFO class*, which correspond to abstract concepts imported from the Basic Formal Ontology, except for *role* and *occurrence*, which we add as subclasses of the root *entity*. By performing a depth-first search (DFS) from the root entity, we break any cycle, bypass any class without description, and remove any node that is not a transitive subclass of *entity*.

[2]https://www.wikidata.org/wiki/Help:Property_constraints_portal
[3]https://www.wikidata.org/wiki/Wikidata:WikiProject_Ontology/Cleaning_Task_Force
[4]https://www.wikidata.org/wiki/Wikidata:External_identifiers

(1) Link Checking. To refine the taxonomy, we first question the relevance of each link using LLMs, inspired by [23]. Specifically, if a is a subclass of b in the taxonomy, we ask the LLM to confirm whether a is a subclass of b, given the labels and descriptions of a and b; we also ask the LLM whether b is a subclass of a, in a symmetric way. Using both answers, say the boolean variables $subclass(a, b)$ and $subclass(b, a)$, we deduce the following semantic relation for the link from a to b:

$$\begin{aligned} subclass(a,b) \wedge \neg subclass(b,a) &\Rightarrow confirmed(a,b) \\ subclass(a,b) \wedge subclass(b,a) &\Rightarrow equivalent(a,b) \\ \neg subclass(a,b) \wedge subclass(b,a) &\Rightarrow reversed(a,b) \\ \neg subclass(a,b) \wedge \neg subclass(b,a) &\Rightarrow irrelevant(a,b) \end{aligned}$$

where $confirmed(a, b)$ means that the link is confirmed by the LLM, $equivalent(a, b)$ means classes a and b are equivalent and should be merged, $reversed(a, b)$ means that the link should be reversed, and $irrelevant(a, b)$ means that the link between a and b should be cut. A confidence score is attached to each answer, based on the probability of the token associated with the expected answer (True or False); the predicted semantic relation is taken into account if the answers to both questions are valid (i.e., either True or False) and the minimum of the confidence scores of both answers exceeds some threshold θ. To mitigate bias [7], the final decision is made from majority voting over predictions of several LLMs.

(2) Graph Operations. Given the link predictions of the LLMs, we successively proceed with the following graph transformations. We first (1) cut *irrelevant* links whenever the resulting disconnected subgraph, if any, has at most k nodes, where k is a parameter of the algorithm. We then (2) cut *reversed* links if this does not disconnect any node from the root; otherwise, we merge the two classes. Next, we (3) merge *equivalent* classes and remove the created transitive links, if any. Finally, we (4) iteratively exclude leaf classes that have no instance, and classes of depth at least 3 that do not have a Wikipedia page; disconnected classes are then relinked to their closest parent in the original taxonomy.

Example. An example is shown in Figure 1. The link from *video work* (defined as "work with a video component") to *video recording* (defined as "technique of recording, copying and broadcasting of moving visual images") is predicted as *irrelevant*. This link is then cut, and so eliminates the inconsistent taxonomy path from *film* to *condition*. In addition, *goods and services* is predicted to be equivalent to *product* and is therefore merged into it. Other redundant classes, such as *time-based visual work*, are excluded as they don't have a Wikipedia page. The resulting taxonomy $\mathcal{T}$ is much simpler, more accurate, and less redundant (see Section 4).

3.2 Type Constraints

Next, we address the confusion, redundancy, and orphan types that exist in the type constraints of Wikidata. For each property, we retrieve the declared lists of subject-side and value-side constraint classes and simplify each side independently using our taxonomy $\mathcal{T}$. Specifically, we map each class to its representative in taxonomy $\mathcal{T}$, as induced by the successive merging steps, and drop it otherwise (e.g., if it is a metaclass). We then eliminate redundancy: any class that already has one of its ancestors in the type constraint is dropped. Finally, we simplify the type constraint in three steps: (1) hierarchical clustering of the remaining classes, (2) selection of Lowest Common Ancestors (LCAs) within each cluster, and (3) filtering of orphan types.

(1) Hierarchical Clustering. We cluster the classes of each type constraint as follows. We use the distance between any two classes in the taxonomy $\mathcal{T}$ as a measure of dissimilarity. We then apply agglomerative hierarchical clustering with average linkage on the resulting pairwise dissimilarity matrix and cut the dendrogram at the level corresponding to the largest gap between consecutive merge heights. This cut produces a set of clusters of classes, where each cluster corresponds to classes that are close from one another in the taxonomy $\mathcal{T}$.

(2) Lowest Common Ancestors. For every cluster C, let $\mathcal{L}$ be the set of all Lowest Common Ancestors (LCAs) of non-empty subsets of C in the taxonomy $\mathcal{T}$. We seek to cover C by some minimal subset of elements of $\mathcal{L}$ that are both close to C and do not introduce too many additional instances. For each $\ell \in \mathcal{L}$, let $\mathrm{cov}(\ell) \subset C$ be the subset of C covered by ℓ. For proximity, we keep only those $\ell \in \mathcal{L}$ whose the average distance to their covered set $\mathrm{cov}(\ell)$ in the taxonomy $\mathcal{T}$ is less than some parameter d. For representativeness, inspired by [25], we keep only $\ell \in \mathcal{L}$ whose proportion of additional cumulative instances (i.e., cumulative instances of ℓ that are not cumulative instances of $\mathrm{cov}(\ell)$) is less than some parameter τ. Finally, we select a set $\mathcal{S}$ of minimum size that covers C, i.e., $\cup_{\ell \in \mathcal{S}} \mathrm{cov}(\ell) = C$, among the remaining elements of $\mathcal{L}$.

(3) Orphan Types. Finally, we remove orphan types from the constraint, i.e., any type that is not instantiated for this property after importing all valid facts (see Section 3.3).

Example. Consider the subject-type constraint of property *notable work* in Figure 1. The classes *human* and *animal actor* are clustered to their LCA, *individual animal*, thereby resolving the subject-type constraint violation. Other classes, such as *group of humans* (a metaclass) or *artificially intelligent entity* (an orphan type), are removed. The class *character* is also removed as it is already covered by its ancestor *imaginary character*, which was merged into the class *fictional entity* in the step of taxonomy cleaning.

3.3 Fact Cleaning

Finally, we import facts from Wikidata using the cleaned taxonomy and type constraints. We proceed in two steps.

(1) Instance Typing. We collect instances and their type facts from Wikidata, based on the *instanceOf* (P31) property. We discard all type facts whose objects are not classes of the cleaned taxonomy $\mathcal{T}$. If the class has been filtered out in step (4) of the graph operations, we retype its instance to its closest ancestor, unless it is the root *entity*. To reduce redundancy, we also remove all type facts that can be deduced from another type fact by transitivity.

(2) Constraint Filtering. We then import all facts that meet the subject or value-type constraints derived above. Specifically, a triple (s, p, o) whose property p has either a subject or object type constraint is imported whenever at least one direct type of the subject s is a transitive subclass of one of the admissible subject classes

Table 1: Intrinsic Evaluation

Criterion	Metric	Wikidata	WiCleanData
Memory	Dump size	985G	**19G**
Complexity	Classes	4.2M	**35K**
	Top-level classes	30	**18**
	Metaclasses	38K	**0**
	Depth of the taxonomy	22	**14**
	Avg. number of paths to root	75	**5**
	Avg. subject type constraints	2.5	**1.8**
	Avg. value type constraints	3.6	**2.4**
Conciseness	Taxonomic loops	36	**0**
	Redundant taxonomic links	511K	**0**
	Classes without instances	93%	**3%**
Readability	Entities without label	24%	**0%**
Coverage	Classes per instance	**46.8**	13.2
	Facts per instance	**21.3**	5.3
Robustness	Taxonomy robustness	**0.80**	0.75

of p, and likewise for the object o. Those facts whose properties have no type constraints are also imported.

Results. The resulting knowledge graph, WiCleanData, has around 50M entities and 420M facts, representing about 100% and 93%, respectively, of those extracted from Wikidata (excluding scholarly articles, in particular). In other words, we retain nearly all entities but exclude 7% of the facts due to missing or incorrect types. By design, all facts of WiCleanData satisfy the type constraints.

The source code[5] is publicly available for reproducibility. The automated pipeline was applied using three LLMs of different groups (Mistral-Small-24B, Qwen3-32B, and Gemma-3-27B) and the following parameters: $\theta = 0.5, k = 3, d = 2, \tau = 0.02$. A Web interface[6] is provided to facilitate exploration of the knowledge graph. It includes a SPARQL query endpoint, tools to compare the taxonomy to that of Wikidata and to explain how the type constraints are generated for each property.

4 Evaluation

We now assess the quality of WiCleanData from both intrinsic and extrinsic perspectives.

Intrinsic evaluation. Following [27, 31], we evaluate WiCleanData in terms of complexity, conciseness, readability, coverage, and robustness. The results are given in Table 1. As expected, WiCleanData has a simpler, more concise taxonomy and fewer constraint types than Wikidata. It also has fewer classes per instance, as redundancy has been reduced. The larger number of facts per instance is mainly due to instances of *scholarly article* and facts with external identifiers, which we have filtered out. We note that some classes in WiCleanData still have no direct instances. These classes are typically in the upper part of taxonomy, such as *spatial boundary*, having subclasses like *coastline* with instances. For robustness, we use the metric in [31] that reflects the ability of a taxonomy to distinguish between different instances. It is typically inversely related to conciseness, as a large number of classes makes entities easier to discriminate. We observe that WiCleanData nearly preserves the taxonomic robustness, while substantially reducing redundancy.

[5]https://github.com/peng-yiwen/wicleanData
[6]https://wicleandata.r2.enst.fr/

Extrinsic evaluation. We further evaluate WiCleanData on multiple downstream tasks to show its usefulness and coverage. Specifically, we use the KGrEaT [12] framework, which integrates multiple datasets and evaluation metrics into a modular architecture to evaluate the extrinsic utility of KGs. The results are shown in Table 2. WiCleanData achieves comparable or even better performance than Wikdiata on all tasks of different types, proving its usability and coverage despite removing a large number of classes and facts. Clustering performance is slightly lower in terms of ARI and NMI, likely due to the removal of metaclass and their instances (see Limitations (1)), which reduces minority-cluster detectability.

Table 2: Extrinsic Evaluation (PK = Precision for known entities, PA = Precision for all entities).

Task Type	Metric	Wikidata		WiCleanData	
		PK	PA	PK	PA
Classification	Accuracy ↑	**0.558**	0.344	0.551	**0.408**
Regression	RMSE ↓	0.648	1.243	**0.645**	**1.219**
Clustering	ARI ↑	**0.197**	0.086	0.114	**0.087**
	NMI ↑	**0.229**	**0.154**	0.115	0.147
	Accuracy ↑	0.621	0.318	**0.724**	**0.426**
Doc. Sim.	Spearman ↑	0.167	0.108	**0.200**	**0.199**
	Pearson ↑	0.258	0.126	**0.273**	**0.270**
	Harm. Mean ↑	0.197	0.116	**0.230**	**0.228**
Recommend.	F1 ↑	**0.013**	**0.001**	**0.013**	**0.001**

Limitations. (1) In this work, we remove all metaclasses to simplify the taxonomy, which leads to the loss of some specific classes (e.g., *type of chemical entity*, *class of anatomical entity*). In Wikidata, instances of these classes are not always classes (e.g., *Celahin D*, which is a molecule). Note that the concept of class order in Wikidata is controversial and difficult for most contributors to understand and refine [5, 20]. (2) While LLMs show high reliability in hierarchy discovery [28], they remain error-prone. Currently, our predictions rely on a single prompt template, though more effective templates may exist. Instead of exploring various combinations of LLMs and prompt templates, an alternative approach is to use confidence scores and study the taxonomy sensitivity to some confidence threshold. (3) Current LLM prompting considers only local subclass links and ignores long-path dependencies. For instance, *book* is correctly considered as a subclass of *communications media*, but is arguably not a transitive subclass of *state*.

5 Conclusion

In this paper, we introduce WiCleanData, a refined version of Wikidata containing a clean taxonomy, simplified type constraints, and type-consistent facts. It is built via an automatic three-stage pipeline: LLM-based taxonomy refinement, type constraint cleaning via hierarchical aggregation, and constraint-based fact filtering for type consistency. Intrinsic evaluation shows that WiCleanData is less complex, more concise, and remains taxonomically robust. Extrinsic evaluation uses WiCleanData as background knowledge for multiple downstream tasks, and shows that it improves performance over Wikidata, confirming its usefulness and coverage. As for future work, it includes several directions. For example, improve metaclass management [5] and consider more advanced repair strategies such as weakening and completion[16].

GenAI Usage Disclosure

As described in Section 3.1, we adopted generative AI as part of the taxonomy refinement pipeline to assess the semantic validity of taxonomy links. We also used ChatGPT for language editing to improve the clarity of the manuscript. GenAI tools (e.g., Claude Code) were additionally used to assist with writing a small portion of the code, which was carefully reviewed, tested, and verified by the authors. Beyond these uses, GenAI tools were not used for experimental results, scientific claims, or performance analysis. The authors take full responsibility for the content of this paper.

References

[1] David Abián, Francesco Guerra, J Martínez-Romanos, and Raquel Trillo-Lado. 2017. Wikidata and DBpedia: a comparative study. In Semanitic keyword-based search on structured data sources. Springer, 142–154.

[2] Abdallah Arioua and Angela Bonifati. 2018. User-guided repairing of inconsistent knowledge bases. In EDBT 2018-21st International Conference on Extending Database Technology. OpenProceedings. org, 133–144.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In International Semantic Web Conference. Springer, 722–735.

[4] Moria Bergman, Tova Milo, Slava Novgorodov, and Wang-Chiew Tan. 2015. QOCO: A query oriented data cleaning system with oracles. Proceedings of the VLDB Endowment 8, 12 (2015), 1900–1903.

[5] Freddy Brasileiro, João Paulo A Almeida, Victorio A Carvalho, and Giancarlo Guizzardi. 2016. Applying a multi-level modeling theory to assess taxonomic hierarchies in Wikidata. In Proceedings of the 25th international conference companion on World Wide Web. 975–980.

[6] Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 6101–6119.

[7] Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024. Are more llm calls all you need? towards the scaling properties of compound ai systems. Advances in Neural Information Processing Systems 37 (2024), 45767–45790.

[8] Na Dong, Natthawut Kertkeidkachorn, Xin Liu, and Kiyoaki Shirai. 2025. Refining noisy knowledge graph with large language models. In Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK). 78–86.

[9] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. 2017. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. Semantic Web 9, 1 (2017), 77–129.

[10] Nicolas Ferranti, Dayane Guimarães, Jairo Francisco de Souza, and Axel Polleres. 2025. Formalizing Repairs for Wikidata Constraint Violations: A Taxonomy and Empirical Analysis. In International Semantic Web Conference. Springer, 369–387.

[11] Congcong Ge, Yunjun Gao, Honghui Weng, Chong Zhang, Xiaoye Miao, and Baihua Zheng. 2020. Kgclean: An embedding powered knowledge graph cleaning framework. arXiv preprint arXiv:2004.14478 (2020).

[12] Nicolas Heist, Sven Hertling, and Heiko Paulheim. 2023. KGrEaT: a framework to evaluate knowledge graphs via downstream tasks. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. 3938–3942.

[13] Shengbin Jia, Yang Xiang, Xiaojun Chen, and Kun Wang. 2019. Triple trustworthiness measurement for knowledge graph. In The World Wide Web Conference. 2865–2871.

[14] Linnan Jiang, Lei Chen, and Zhao Chen. 2018. Knowledge base enhancement via data facts and crowdsourcing. In 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE, 1109–1119.

[15] Furong Li, Xin Luna Dong, Anno Langen, and Yang Li. 2017. Knowledge verification for long-tail verticals. Proceedings of the VLDB Endowment 10, 11 (2017), 1370–1381.

[16] Ying Li and Patrick Lambrix. 2024. Repairing networks of EL⊥ ontologies using weakening and completing. In International Semantic Web Conference. Springer, 107–125.

[17] Jiaqing Liang, Yanghua Xiao, Yi Zhang, Seung-won Hwang, and Haixun Wang. 2017. Graph-based wrong isa relation detection in a large-scale lexical taxonomy. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31.

[18] André Melo and Heiko Paulheim. 2017. An approach to correction of erroneous links in knowledge graphs. In CEUR Workshop Proceedings, Vol. 2065. RWTH Aachen, 54–57.

[19] Andre Melo and Heiko Paulheim. 2020. Automatic detection of relation assertion errors and induction of relation constraints. Semantic Web 11, 5 (2020), 801–830.

[20] Peter F Patel-Schneider and Ege Atacan Doğan. 2024. Class Order Disorder in Wikidata and First Fixes. arXiv preprint arXiv:2411.15550 (2024).

[21] Heiko Paulheim. 2016. Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web 8, 3 (2016), 489–508.

[22] Thomas Pellissier Tanon, Camille Bourgaux, and Fabian Suchanek. 2019. Learning how to correct a knowledge base from the edit history. In The World Wide Web Conference. 1465–1475.

[23] Yiwen Peng, Thomas Bonald, and Mehwish Alam. 2024. Refining wikidata taxonomy using large language models. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. 5395–5399.

[24] Jonathan Raiman and Olivier Raiman. 2018. Deeptype: multilingual entity linking by neural type system evolution. In Proceedings of the AAAI conference on artificial intelligence, Vol. 32.

[25] Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy (IJCAI'95). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 448–453.

[26] Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro Szekely. 2022. A study of the quality of Wikidata. Journal of Web Semantics 72 (2022), 100679.

[27] Fabian M Suchanek, Mehwish Alam, Thomas Bonald, Lihu Chen, Pierre-Henri Paris, and Jules Soria. 2024. Yago 4.5: A large and clean knowledge base with a rich taxonomy. In Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval. 131–140.

[28] Yushi Sun, Hao Xin, Kai Sun, Yifan Ethan Xu, Xiao Yang, Xin Luna Dong, Nan Tang, and Lei Chen. 2024. Are Large Language Models a Good Replacement of Taxonomies? Proceedings of the VLDB Endowment 17, 11 (2024), 2919–2932.

[29] Gerald Töpper, Magnus Knuth, and Harald Sack. 2012. DBpedia ontology enrichment for inconsistency detection. In Proceedings of the 8th International Conference on Semantic Systems. 33–40.

[30] Hai Dang Tran and Andrew Yates. 2022. Dense retrieval with entity views. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 1955–1964.

[31] Michael Unterkalmsteiner and Waleed Abdeen. 2023. A compendium and evaluation of taxonomy quality attributes. Expert Systems 40, 1 (2023), e13098.

[32] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledge base. Commun. ACM 57, 10 (2014), 78–85.

[33] Chengyu Wang, Rong Zhang, Xiaofeng He, and Aoying Zhou. 2016. Error link detection and correction in wikipedia. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. 307–316.

[34] Chao Wu, Zeyu Zeng, Yajing Yang, Mao Chen, Xicheng Peng, and Sannyuya Liu. 2023. Task-driven cleaning and pruning of noisy knowledge graph. Information Sciences 646 (2023), 119406.