Enriching Wikidata with Linked Open Data

Bohui Zhang[1], Filip Ilievski[1] and Pedro Szekely[1]

[1]*Information Sciences Institute, University of Southern California*

Abstract
Large public knowledge graphs, like Wikidata, provide a wealth of knowledge that may support real-world use cases. Yet, practice shows that much of the relevant information that fits users' needs is still missing in Wikidata, while current linked open data (LOD) tools are not suitable to enrich large graphs like Wikidata. In this paper, we investigate the potential of enriching Wikidata with structured data sources from the LOD cloud. We present a novel workflow that includes gap detection, source selection, schema alignment, knowledge retrieval, and semantic validation. We evaluate our enrichment method with two complementary LOD sources: a noisy source with broad coverage, DBpedia, and a manually curated source with a narrow focus on the art domain, Getty. Our experiments show that our workflow can enrich Wikidata with millions of novel statements from external LOD sources with high quality. Property alignment and data quality are key challenges, whereas entity alignment and source selection are well-supported by existing Wikidata mechanisms.

1. Introduction

Wikidata [1], the largest public knowledge graph (KG), has nearly 1.5B statements about 90M entities. This breadth of information inspires many use cases: museum curators can use Wikidata to describe their art collections comprehensively, while movie critics could quickly query and aggregate statistics about recent movies, and analyze them based on their genre. However, while Wikidata's data model allows for this information to be present, practice shows that much of the relevant information is still missing. For example, around half of the artists in Wikidata have a date of birth, and only 1.88% of the movies recorded in 2020 have information about their cost. Thus, if a user wants to analyze the cost of the films produced in 2020, Wikidata will provide cost for only 60 out of its 2,676 recorded films. This triggers *hunger for knowledge* [2], activating a goal for the user to seek the missing information. As this information is unlikely to be found in an aggregated form, and gathering information about thousands of films is tedious, one must rely on automated tools that can enrich Wikidata with relevant information. Such information might be available in external linked open data (LOD) sources like DBpedia [3] or LinkedMDB [4]; yet, no existing methods can enrich Wikidata with LOD information. Conversely, link prediction [5] accuracy is not sufficient to use it to impute missing information with representation learning.

The traditional LOD workflow includes declarative language tools for schema mapping and templated rewritting of CSV files into RDF [6, 7, 8]. Subsequent ontology alignment [9] can be employed to discover `owl:sameAs` links between the two datasets. Yet, merely collapsing nodes

Wikidata'22: Wikidata workshop at ISWC 2022
bohuizha@usc.edu (B. Zhang); ilievski@isi.edu (F. Ilievski); pszekely@isi.edu (P. Szekely)

Table 1
Example statements found in DBpedia.

DBpedia statement	Wikidata mapping	Assessment
`dbr:Lesburlesque dbp:genre dbr:Burlesque`	`Q6530279 P136 Q217117`	Correct
`dbr:Amanda_de_Andrade dbp:position 'Left back'en`	`Q15401730 P413 'Left back'en`	Wrong datatype
`dbr:Diego_Torres_(singer) dbp:genre dbr:Flamenco`	`Q704160 P2701 Q9764`	Wrong semantic type
`dbr:Eternal_Moment dbp:director dbr:Zhang_Yibai`	`Q5402674 P4608 Q8070394`	Logical, inaccurate

based on `owl:sameAs` relations is not sufficient, as this may merge related but dissimilar nodes (e.g., Barack Obama and the US government) [10]. Schema alignment between two sources has been the main focus so far, but it may be less critical when enriching Wikidata, which provides 6.8k external ID properties that explicitly link to entities in other sources.[1] A key challenge for Wikidata is data quality [11], as external knowledge may be noisy, contradictory, or inconsistent with existing knowledge in Wikidata. Table 1 presents candidate enrichment statements retrieved from DBpedia: only one out of four statements is correct, whereas the other three have an incorrect datatype, semantic type, or veracity. It is unclear how to employ principled solutions for large-scale, high-quality enrichment of Wikidata.

In this paper, we investigate *how to enrich Wikidata with freely available knowledge from the LOD cloud.* We introduce a novel LOD workflow to study the potential of the external identifier mechanism to facilitate vast high-quality extensions of Wikidata. We start by aligning the entities automatically through external identifiers, and we infer the property in the external source based on structural and content information. This schema alignment yields candidate statements, whose quality is rigorously checked through datatype comparison, semantic constraints, and custom validators. We assess the effectiveness of the enrichment workflow on two LOD knowledge sources: Getty [12], a manually curated domain-specific KG with a narrow focus on art, and DBpedia [3], a broad coverage KG which has been automatically extracted from Wikipedia. Extensive experiments on these sources show that our method facilitates vast and high-quality extension of Wikidata with missing knowledge.

We make our code and data available to facilitate future work.[2]

2. Related work

Schema mapping languages, like R2RML [6], Karma [7], and RML [8] enable users to map relational databases into RDF, optionally followed by a semi-automatic ontology alignment step. Recent ontology alignment methods [9] typically align two ontologies in the vector space.

[1]https://www.wikidata.org/wiki/Category:Properties_with_external-id-datatype, accessed March 2, 2022.
[2]https://github.com/usc-isi-i2/hunger-for-knowledge

Ontology enrichment deals with noise, incompleteness, and inconsistencies of ontologies, by discovering association rules [13], or by extracting information from WWW documents [14]. Ontology evolution seeks to maintain an ontology up to date with respect to the domain that it models, or the information requirements that it enables [15].

Prior work has attempted to enrich Wikidata with satellite data for a given domain, including frame semantics [16], biodiversity data [17, 18], and cultural heritage data [19]. The community's commonly adopted way to automatically edit Wikidata is to develop bots that inject the knowledge from the external source into Wikidata [17, 18]. By learning how to represent entities in a KG like Wikidata, link prediction models [5] can predict missing values by associating them to known statements. Our work complements prior efforts that enrich Wikidata with domain-specific data or predict missing links, as we aim to devise a method for generic, large-scale enrichment of Wikidata with external LOD sources.

Identity links in the LOD cloud could be explored to combine information from different sources. However, as we show in this paper, identity links between entities are insufficient for KG enrichment, as they do not account for the quality and the compatibility of the data. In that sense, our work relates to prior work that explores identity links in the LOD cloud [10] or devises mechanisms to discover latent identity links [20]. The goal of our work is different - to enrich Wikidata by finding high-quality knowledge in well-connected sources.

The automatic validation of our method relates to efforts that analyze the quality of large KGs. Beek et al. [21] propose a framework for analyzing the quality of literals on the LOD Laundromat [22], but it is unclear how to generalize this framework to entity nodes. Prior work has studied the quality of Wikidata [23] and compared it to the quality of other KGs, like YAGO [24] and DBpedia [25]. Shenoy et al. [26] apply five semantic constraints on Wikidata in order to measure statement quality. None of these works has investigated how to automatically validate external statements that can be used to enrich Wikidata.

Wikidata's property constraint pages define existing property constraints and report number of violations for a single dump.[3] Recoin [27] computes relative completeness of entity information by comparing the available information for an entity against other similar entities. Objective Revision Evaluation Service (ORES),[4] provides AI-based quality scores (e.g., for vandalism) for revisions in Wikidata. Our work complements these efforts by Wikidata, by providing mechanisms for automatic alignment and semantic validation of knowledge from the LOD cloud before it is submitted to Wikidata.

3. Method

Our enrichment method is shown in Figure 1. Given a user query, our method queries Wikidata (W), obtaining a set of statements $S_w = \{s_w = (e_w, p, o_w) \mid e_w \in E_w\}$ for known subjects E_w, and a set of subjects E_u with unknown values, for industry in the example. We call this step *gap detection*, as it generates a set of entities for which we seek missing knowledge in order to satisfy the user's query needs. Considering that the LOD cloud contains other sources that are likely to contain information about the same entities, we perform a manual *KG selection*

[3]https://www.wikidata.org/wiki/Help:Property_constraints_portal
[4]https://www.wikidata.org/wiki/Wikidata:ORES

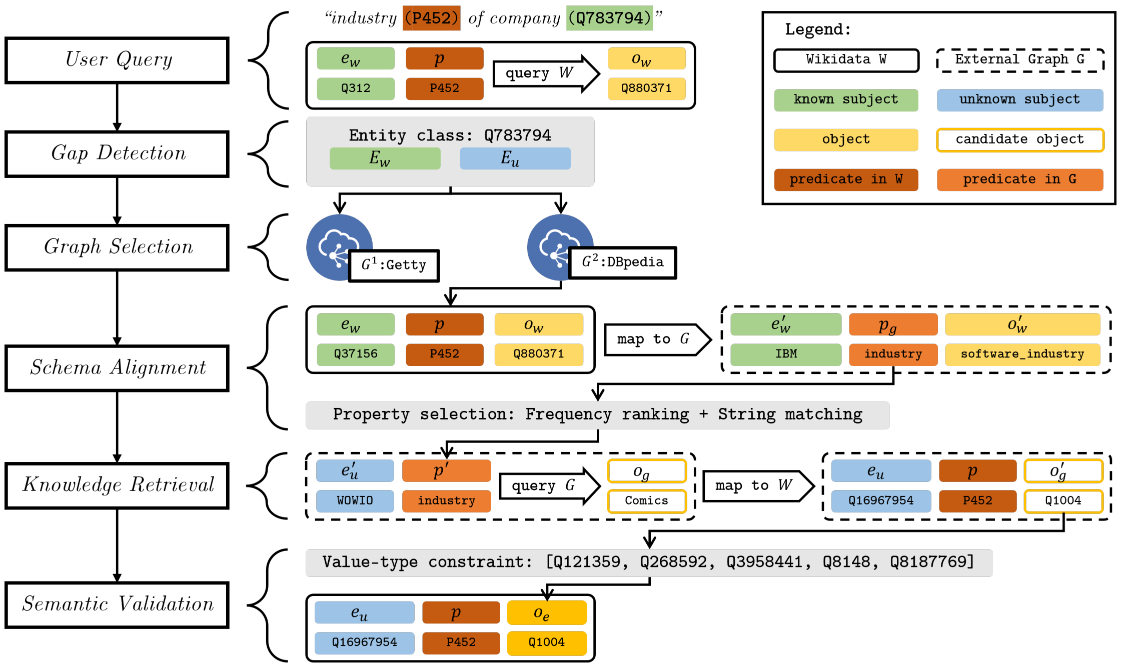


Figure 1: Our enrichment method, illustrated on enriching Wikidata with additional knowledge from DBpedia about the query "industry of companies".

step to determine a relevant set of KGs (G) to consult for the entities in E_u. Here, the sources G (DBpedia in this example) are assumed to overlap with W in terms of the entities they describe. Our *schema alignment* step consolidates the entities and properties of W with those of each G, since their entity and property identifiers are generally different. Each pair (e_w, o_w) is aligned to a DBpedia-valued pair $(e_w{}', o_w{}')$. Similarly, the unknown entities $e_u \in E_u$ are mapped in the same way to DBpedia entities $e_u{}'$. Then, the external KG DBpedia is queried for property paths which correspond to the known subject-object pairs $(e_w{}', o_w{}')$. In the set of DBpedia property p_g, path p' (dbp:industry) that corresponds to the Wikidata property p is discovered with our method. After aligning the two schemas, the knowledge retrieval step obtains values from DBpedia, by querying for $(e_u{}', p')$ pairs comprised of DBpedia entities and property paths, resulting in a set of newly found statements $S_g = \{s_g = (e_u{}', p', o_g)\}$. A *semantic validation* step is employed in order to ensure that the semantics of the newly found statements in S_g after mapping back to W correspond to the semantics intended by Wikidata. The set of validated statements $S_e = \{s_e = (e_u, p, o_e)\} \subseteq S_g$ is finally used to enrich W. This procedure yields a more complete response to the user query consisting of a union of the original and the enriched statements, formally: $S_{total} = S_w \cup S_e$.

3.1. Gap Detection

We consider a structured query against a target knowledge graph W for a query property p (e.g., industry). The gap detection step generates a set of subject entities E_w for which the value for the property p is known in Wikidata, and a set of entities E_u for which the value of the property p is missing in W. E_w and E_u are subsets of the overall set of target entities and they are mutually disjoint, formally: $E_w \subseteq E$, $E_u \subseteq E$, $E = E_k \bigcup E_u$, and $E_w \bigcap E_u = \emptyset$.

This work focuses on finding values for entities in E_u that have zero values for a property in Wikidata. We note that it is possible that the statements in S_w do not fully answer the query for the entities E_w, as these entities may have multiple values for p, e.g., a politician may have several spouses throughout their life. Enriching multi-valued properties will be addressed in future work.

3.2. Graph Selection

The graph selection step manually selects from a set of LOD sources G that can be used to enrich the results of the query. In this work, we consider an automatically extracted general-domain KG (DBpedia) and a domain-specific curated KG (Getty). We experiment with using both KGs, or selecting one based on the posed query.

DBpedia [3] is an open-source KG derived from Wikipedia through information extraction. DBpedia describes 38 million entities in 125 languages, while its English subset describes 4.58 millions of entities. Large part of the content in DBpedia is based on Wikipedia's infoboxes: data structures containing a set of property–value pairs with information about its subject entity, whose purpose is to provide a summary of the information about that subject entity. We use DBpedia infoboxes in order to enrich Wikidata, as this data is standardized, relevant, and expected to be extracted with relatively high accuracy.

Getty [12] is a curated LOD resource with focus on art. In total, Getty contains entities covering 324,506 people and 2,510,774 places. It consists of three structured vocabularies: (1) Art & Architecture Thesaurus (AAT) includes terms, descriptions, and other information (like gender and nationality) for generic concepts related to art and architecture; (2) the Getty Thesaurus of Geographic Names (TGN) has 321M triples with names, descriptions, and other information for places important to art and architecture; and (3) the Union List of Artist Names (ULAN) describes names, biographies, and other information about artists and architects, with 64M statements.

3.3. Schema Alignment

As Getty and DBpedia have a different data model compared to Wikidata, we first align their schemas to Wikidata in order to query them based on missing information in Wikidata. The schema alignment consists of two sequential steps:

1. Entity resolution maps all known subject entities in Wikidata, e_w, unknown subjects e_u, and the known objects o_w to external identifiers e'_w, e'_u, and o'_w, respectively. We map Wikidata nodes to nodes in external KGs automatically, by leveraging external identifiers and sitelinks available in Wikidata. In total, Wikidata contains 6.8K external-id properties. Wikidata contains 46,595,392 sitelinks, out of which 5,461,631 link to the English Wikipedia pages. Our method leverages sitelinks to map Wikidata entities to DBpedia nodes,[5] while for Getty we use vocabulary-specific external-id properties in Wikidata: AAT ID (`P1014`) for AAT items, TGN ID (`P1667`) for TGN items, and ULAN ID (`P245`) for ULAN items. We note that, while the entity mapping is automatic,

[5]Wikipedia page URIs can trivially be translated to DBpedia URIs.

the selection of the external-id property itself in the current method is manual, e.g., the user has to specify that `P1667` should be used for TGN identifiers.

2. Property alignment maps the property p from Wikidata to a corresponding property path p' in G by combining structural and content information. We query G for property paths p_g with maximum length L that connect the mapped known pairs (e'_w, o'_w). We aggregate the obtained results, by counting the number of results for each p_g. We preserve the top-10 most common property paths, and we use string similarity to select the optimal one. Here, we select the top-10 candidates by using Python built-in function based on Gestalt Pattern Matching [28].[6] If the most similar property path has similarity above a threshold (0.9), then we select it as a mapped property p', otherwise we select the top-1 most frequent property. In the example in Figure 1, the Wikidata property `P452` would map to `dbp:industry` in DBpedia, which is both the top-1 most frequent property in the aligned results, and it has maximum string similarity with p. We expect that string similarity and value frequency can complement each other. For example, the top-1 most frequent property for `P149` (architectural style) is `dbp:architecture`, while string similarity filter the right mapping `dbp:architecturalStyle` from the candidates. The property chain p' can be quite complex, e.g., Wikidata's `place of birth (P19)` maps to a 4-hop path in Getty: `foaf:focus` → `gvp:biographyPreferred` → `schema:birthPlace` → `skos:exactMatch`.

3.4. Knowledge Retrieval

The schema alignment step produces external identifiers for the unknown entities e'_u and an external property path p' that corresponds to the property in the original query. The user can query the external graph G with the mapped subject-property pair (e'_u, p'), in order to automatically retrieve knowledge. In Figure 1, an example of (e'_u, p') pair is (`dbr:WOWIO`, `dbp:industry`). We denote the candidate objects found in G with this step with o_g. As the candidate object identifiers belong to the external graph, the user can perform inverse entity resolution by following sitelinks or external identifiers from the external graph G to W. This step results in newly found Wikidata objects o'_g for the unknown entities e_u, completing their statements (e_u, p, o'_g).

3.5. Semantic Validation

Despite the schema alignment, the candidate objects o'_g may be noisy: they may have a wrong datatype (e.g., date instead of a URI), an incorrect semantic type (e.g., a nationality instead of a country), or a literal value that is out of range (e.g., death year 2042). We trim noisy objects with three validation functions.

1. Datatypes Each Wikidata property has a designated datatype that the candidate objects have to conform to. For instance, the spouses are expected to be Qnodes, while movie costs should be numeric values with units (e.g., 4 million dollars). To infer the expected datatype of a property, we count the datatypes of the known objects o_w, and select the top-1 most common

[6]Our empirical study showed that this function leads to comparable accuracy like Levenshtein distance, but it is more efficient.

datatype. This function returns a subset of statements S_{v1} with candidate objects that belong to the expected datatype.
2. Property constraints We validate the object values further based on property constraints defined in Wikidata. Specifically, we use value-type constraints to validate the semantic type of the objects. Value type constraints are similar to property range constraints [26], but they provide a more extensive definition that includes exception nodes and specifies whether the type property is: `P31` (instance of), `P279` (subclass of), or both. Figure 2 in Appendix shows an example of a value type constraint for the property `P452` (industry). We automatically validate the value type of all statements for a property, by comparing their object value to the expected type. Following [26], we encode the value type constraint for a property as a KGTK [29] query template. Each template is instantiated once per property, allowing for efficient constraint validation in parallel. Constraint violations for a property are computed in a two-step manner: we first obtain the set of statements that satisfy the constraint for a property, and then we subtract this set from the overall number of statements for that property. The constraint validation function yields a set of validated statements S_{v2}.
3. Literal range We validate date properties (e.g., date of birth) by ensuring that they do not belong to the future, i.e., that every recorded date (excluding expected dates) is earlier than the current date. This function outputs a set of valid statements S_{v3}.

The set of validated statements is the intersection of the results returned from the validation functions: $S_v = S_{v1} \cap S_{v2}$ for Qnodes and $S_{v1} \cap S_{v2} \cap S_{v3}$ for date values. This validated statements in S_e have the form (e_u, p, o_e). The total set of statements for the user query becomes $S_{total} = S_w \bigcup S_e$.

4. Experimental Setup

Knowledge graphs We experiment with batch enrichment for 955 Wikidata properties that have value-type constraints. We use the Wikidata 2021-02-15 dump in a JSON format and the 2021-10-27 sitelinks file. We use the 2021-12-01 DBpedia infobox file in a Turtle (.ttl) format.[7] We select its cannonicalized version, as it ensures that its subjects and objects can be mapped to English Wikipedia pages. For the Getty vocabularies, we download the 2021-09-18 dump in N-Triples (.nt) from their website.[8]
Evaluation To evaluate the quality of the overall enrichment, we randomly sample candidate statements and annotate their validity manually. Specifically, we sample 100 statements from the DBpedia set and 30 from Getty. Two annotators first annotate 130 statements independently by searching each one on Internet and identify the correctness, then discuss to resolve two conflicts. In the sampled subset, 20/100 of the DBpedia statements and 11/30 of Getty are correct, while the rest are incorrect. We label three reasons for incorrect statements: wrong datatype, wrong semantic type, and inaccurate information. Out of the 80 incorrect DBpedia statements, 66 have incorrect datatype, 7 incorrect semantic type, and 7 are inaccurate. For Getty, 0 are incorrect datatype, 17 are incorrect semantic type, and 2 are inaccurate.

To investigate the quality of our property alignment, we take property mappings provided by `owl:equivalentProperty` in DBpedia and `P1628` (equivalent property) in Wikidata as ground

[7]https://databus.dbpedia.org/dbpedia/generic/infobox-properties/
[8]http://vocab.getty.edu/

Table 2
Batch enrichment results when using DBpedia, Getty, and both KGs. $|S_*|$ shows numbers of statements. In total, we consider 955 properties. $|p'|$ shows the numbers of properties mapped to each of the KGs.

	$\|p'\|$	$\|S_w\|$	$\|S_g\|$	$\|S_e\|$	$\|S_{total}\|$
DBpedia	582	106,104,551	41,309,864	21,023,187	127,127,738
Getty	3	195,153	10,518	5,766	200,919
Both	582	106,104,551	41,320,382	21,028,953	127,328,657

truth. We dub this data `Equivalence`. In the ground truth pairs we collected, each Wikidata property is mapped to one or multiple DBpedia properties. In total, 88 Wikidata properties are mapped to 101 DBpedia properties. We formulate a task where the goal is to map a Wikidata property to at least one of its corresponding DBpedia properties. Besides correct and incorrect mappings, we annotate an intermediate category of close match for properties that match partially. We show two F1-values of our method: hard, which only counts exact matches, and soft, which includes partial matches. We compare our method to three baselines. The *string matching* and *frequency matching* baselines are ablations of our method that only consider string similarity or frequency, but not both. The third baseline embeds all DBpedia labels with BERT [30] and uses cosine distance to select the closest property label.

5. Results

In this Section, we present five experimental findings that concern the potential of our method, its overall performance per component, and its consistency when the external information is covered by Wikidata.
Finding 1: Our method can enrich Wikidata with millions of statements about millions of entities. Table 2 shows the results of our method for all properties in Wikidata that have a value-type constraint. Out of 955 Wikidata properties, our method is able to align 582 properties with DBpedia and 3 with Getty. For all properties aligned with DBpedia and Getty, we gather 21 million statements enriching the original Wikidata knowledge by 16.54%. Interestingly, while the original Wikidata focuses on a broad coverage of entities, our method is able to enrich more properties for a smaller set of entities, signifying a higher density and a more narrow focus. The $|S_g|$ column shows that our method collects 41 million candidate statements from DBpedia and Getty, out of which 21 million pass the semantic validation. The median number of novel statements per property is 982. For 161 (27.66%) properties, our method provides double or more statements relative to the original set of statements. The relative increase of knowledge is the lowest for the properties `P538` (fracturing), `P209` (highest judicial authority), and `P1283` (filmography). Meanwhile, the properties `P66` (ancestral home), `P500` (exclave of), and `P3179` (territory overlaps) are relatively sparse in Wikidata, and receive many more statements from DBpedia. Comparing the two external KGs, we observe that DBpedia overall contributes many more statements than Getty, and brings a higher enrichment per property, averaging at 16.54%

Table 3
Evaluation results on 130 candidate triples: 100 from DBpedia and 30 from Getty. We show the accuracy, precision, recall and F1-score of our semantic validation on this subset. Getty does not have values with a wrong datatype ('-').

KG	Accuracy	Precision	Recall	F1-score	Accuracy per category			
					Correct	Datatype	Sem. type	Inaccurate
DBpedia	87.00%	61.54%	84.21%	71.11%	80.00%	96.97%	71.43%	28.57%
Getty	93.33%	84.62%	100.0%	91.67%	100.0%	-	100%	0%
Both	88.46%	69.23%	90.00%	78.26%	87.10%	96.97%	91.67%	22.22%

vs 2.87% for Getty.
Finding 2: The overall quality of the enriched statements is relatively high. Table 3 shows that our method can distinguish between correct and incorrect statements with a relatively high accuracy of over 88%. As the majority of the candidates are incorrect, we observe that the F1-score is lower than the accuracy. The precision of our method is lower than the recall on both DBpedia and Getty, which indicates that most of the disagreement of our method with human annotators is because of false positives, i.e., incorrect statements identified as correct. This indicates that our semantic validation is accurate but it is not complete, and it could benefit from additional validators. This observation is further supported by the relatively higher precision and recall of our method on Getty in comparison to DBpedia. As Getty is manually curated and enforces stricter semantics, it has a smaller range of data quality aspects to address, most of which are already covered by our method. The quality issues in the case of DBpedia are heterogeneous, as a result of its automatic extraction and lack of curation. We find that out of 130 triples, 52 had incorrect property mappings, and the semantic validation is able to correct 44 of them. For example, Wikidata property `P208` (executive body) got mapped to `dbp:leaderTitle` in DBpedia, and its value `Q30185` was not allowed by the Wikidata constraint for `P208`. We evaluate property alignment and semantic validation in more detail later in this Section.
Finding 3: Property mapping performance is relatively high, but sparse properties are difficult. Table 2 showed that around 40% of the target Wikidata properties had no match found in DBpedia with our method. To investigate whether this is because of misalignment between the two schemas or a limitation of our method, we evaluate the property alignment step on the `Equivalence` data. The results are shown in Table 4. Our method achieves the best F1-score for both the soft match cases (89%) and the hard match cases (66%).[9] As frequency and string matching are ablations from our method, their lower performance supports our decision to combined them to get the best of both worlds. For instance, frequency matching tends to prefer more general properties over specific ones, mapping `P30` (continent) to `dbp:location`. Thanks to string matching, our method predicts the right property `dbp:continent` in this case. Conversely, string matching is easily confused by cases where the labels are close but the

[9]We also manually evaluate the property matching methods on a separate randomly chosen set of 20 properties, and observe similar results.

Table 4
Evaluation results on known aligned properties between DBpedia and Wikidata. We compare against exact match and language model baselines. We also show the performance of our method per property quartile, where the quartiles are based on the number of examples for a property.

Method	Hard match			Soft match		
	Precision	Recall	F1-score	Precision	Recall	F1-score
BERT Embedding	47.73%	47.73%	47.73%	72.73%	72.73%	72.73%
Frequency Matching	52.33%	51.14%	51.72%	81.40%	79.55%	80.46%
String Matching	52.27%	52.27%	52.27%	86.36%	86.36%	86.36%
Our Method	**66.28%**	**64.77%**	**65.52%**	**89.53%**	**87.50%**	**88.51%**
Our method (Q1)	71.43%	68.18%	69.77%	90.48%	86.36%	88.37%
Our method (Q2)	63.64%	63.64%	63.64%	100.0%	100.0%	100.0%
Our method (Q3)	81.82%	81.82%	81.82%	86.36%	86.36%	86.36%
Our method (Q4)	47.62%	45.45%	46.51%	80.95%	77.27%	79.07%

actual meaning is not related, e.g., it maps `P161` (cast member) to `dbp:pastMember`, whereas our method correctly maps it to the ground truth result `dbp:starring` owing to its frequency component. As our method still largely relies on the frequency of statements, we hypothesize that its performance decreases for properties with fewer known statements. To investigate this hypothesis, we divide the ground truth properties into four quartiles (of 22 properties) based on the descending size of their original Wikidata statements. We evaluate the accuracy or our property matching per quartile. We note that the performance of the first three quartiles with larger number of statements is relatively better than the last quartile, which indicates that the precision of our method is positively correlated with the size of known statements. This limitation can be addressed in the future with more robust methods, e.g., based on learned property representations.

Finding 4: Semantic validation can detect wrong datatypes and semantic types. Table 2 shows that the semantic validation has a large impact on the results: out of 41 million candidate statements initially found by our method, around half of them satisfy our validation function. The compatibility ratios are similar for both Getty and DBpedia (50.89% to 54.82%), which is surprising, considering that DBpedia has been largely extracted automatically and is error-prone, whereas Getty is well-curated and considered an authority. To study the precision and recall of our semantic validation, we annotate three reasons for incorrect statements: wrong datatype, wrong semantic type, and inaccurate information. We found that (Table 2) our method performs well on identifying correct statements (F1-score 93.10%), as well as detecting errors due to wrong datatypes (F1-score 96.97%) and incorrect semantic types (F1-score 91.67%). It performs relatively worse when the statements satisfy the validation but are inaccurate. For example, the enriched statement (`Q6712846 P19 Q49218`) for the property `P19` (place of birth) from Getty is logical but inaccurate, since the value `Q49218` satisfies the value-type constraints while it is not the actual birth place of `Q6712846`. Among 130 triples, our method produces 7 false positive cases that are factually incorrect. These results are expected, given that our method is designed

Table 5
Evaluation results for data consistency. Numbers of Wikidata statements, enriched statements, overlapping entity-property values, agreeing statements, and disagreeing statements are counted. The agreement ratio r_{agree} is calculated by $|S_{agree}|/|S_{overlap}|$.

KG (Property)	$\lvert S_w\rvert$	$\lvert S_e\rvert$	$\lvert S_{overlap}\rvert$	$\lvert S_{agree}\rvert$	$\lvert S_{disagree}\rvert$	r_{agree}
DBpedia (P19)	2,711,621	467,976	884,078	461,089	422,989	52.15%
DBpedia (P20)	1,080,900	119,161	219,447	128,523	90,924	58.57%
Getty (P19)	65,411	2,939	16,304	13,607	2,697	83.46%
Getty (P20)	50,295	2,556	14,722	12,594	2,128	85.55%

to filter out illogical information, while analyzing veracity is beyond its current scope.
Finding 5: Most results for functional properties are consistent between external graphs and Wikidata, many disagreements are due to different granularities. The analysis so far focused on the novel statements, i.e., provide novel object values for subject-property pairs that do not have them in Wikidata. Here, we measure consistency between the validated statements of the known entities from the external graph with the known statements from Wikidata. We select two functional properties: P19 (place of birth) and P20 (place of death), and count the agreements and disagreements for both DBpedia and Getty. The results in Table 5 for P19 and P20 show that 52-59% of the overlapping statements between Wikidata and DBpedia, and 83-86% of the overlapping statements between Wikidata and Getty coincide. Qualitative inspection of the disagreements reveals that many of the disagreements are due to different granularity choices between the two graphs. For instance, in Wikidata, the place of death of Q1161576 (Daniel Lindtmayer II) is Q30978 (Central Switzerland) which is a region, while Getty provides the specific city Q4191 (Lucerne).

6. Discussion and Future Work

Our enrichment method has been shown to quickly retrieve millions of novel property values in the LOD cloud for entities in Wikidata. As some of the LOD knowledge is extracted in an automatic way, ensuring quality is important - our semantic validation based on datatypes and constraints found around half of the candidate statements to be invalid. Analysis of a subset of the enriched statements revealed that the accuracy of our method is close to 90%, which is reasonably high. Still, our method is merely a step towards the ambitious goal of addressing the notorious challenge of sparsity of today's large KGs [31]. Here, we discuss three key areas of improvement for our method:
1. Enrichment with more LOD KGs - We showed that our method is effective with two external KGs: a general-domain and automatically extracted knowledge graph, DBpedia, and a domain-specific, well-curated knowledge graph, Getty. As Wikidata is still largely incomplete after this enrichment, we can use the 6.8k external identifier properties provided by Wikidata to enrich with thousands other sources. While we expect that our method can be applied on

these thousands of sources, an in-depth investigation of the potential and the quality of this enrichment is beyond the scope of the current paper.
2. Semantic validation - Our method validates candidate statements through datatype and value type constraints. Value type constraints ensure semantic type compatibility of the retrieved statements, yet they are only one of the 30 property constraint types defined in Wikidata. Other Qnode constraints in Wikidata can be employed to generalize our method. For instance, Qnode-valued statements can be further validated via constraints like `one-of (Q21510859)`, whereas literals can be checked with `range (Q21510860)`.
3. Validating veracity Table 3 shows that our method performs relatively well at detecting statements with incorrect datatype or semantic type, whereas it is usually unable to detect inaccurate statements. As discussed by Piscopo [11], veracity is a key aspect of quality of knowledge in KGs. Our method can be further enhanced with models that detect KG vandalism [32] or estimate trust of sources (e.g., through references) [23] to estimate veracity.

7. Conclusions

Recognizing the notorious sparsity of modern knowledge graphs, such as Wikidata, and the promise of linked data information, like external identifiers, to facilitate enrichment, we proposed a method which consisted of five steps (gap detection, external graph selection, schema alignment, knowledge retrieval, and semantic validation) for enriching Wikidata with external KGs found in the LOD cloud. Our experiments showed that the LOD-based method can enrich Wikidata with millions of new high-quality statements from DBpedia and Getty. High-quality enrichment is achieved based on large-scale automated semantic validation and a hybrid algorithm for property alignment. A key future direction is evaluating the generalization of our method on thousands of LOD sources that Wikidata points to, which opens novel challenges of source selection, more extensive semantic validation, and trust.

References

[1] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.
[2] F. Ilievski, P. Vossen, M. Van Erp, Hunger for contextual knowledge and a road map to intelligent entity linking, in: International Conference on Language, Data and Knowledge, Springer, Cham, 2017, pp. 143–149.
[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The semantic web, Springer, 2007, pp. 722–735.
[4] O. Hassanzadeh, M. P. Consens, Linked movie data base, in: LDOW, 2009.
[5] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26 (2013).
[6] S. Das, R2rml: Rdb to rdf mapping language, http://www. w3. org/TR/r2rml/ (2011).
[7] C. A. Knoblock, P. Szekely, J. L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea,

M. Taheriyan, P. Mallick, Semi-automatically mapping structured sources into the semantic web, in: Extended Semantic Web Conference, Springer, 2012, pp. 375–390.

[8] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, Rml: a generic language for integrated rdf mappings of heterogeneous data, in: Ldow, 2014.

[9] S.-C. Chu, X. Xue, J.-S. Pan, X. Wu, Optimizing ontology alignment in vector space, Journal of Internet Technology 21 (2020) 15–22.

[10] W. Beek, J. Raad, J. Wielemaker, F. Van Harmelen, sameas. cc: The closure of 500m owl: sameas statements, in: European semantic web conference, Springer, 2018, pp. 65–80.

[11] A. Piscopo, E. Simperl, What we talk about when we talk about wikidata quality: a literature survey, in: Proceedings of the 15th International Symposium on Open Collaboration, 2019, pp. 1–11.

[12] P. Harpring, Development of the getty vocabularies: Aat, tgn, ulan, and cona, Art Documentation: Journal of the Art Libraries Society of North America 29 (2010) 67–72.

[13] C. d'Amato, S. Staab, A. G. Tettamanzi, T. D. Minh, F. Gandon, Ontology enrichment by discovering multi-relational association rules from ontological knowledge bases, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016, pp. 333–338.

[14] A. Faatz, R. Steinmetz, Ontology enrichment with texts from the www, Semantic Web Mining 20 (2002).

[15] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, M. Sabou, Ontology evolution: a process-centric survey, The knowledge engineering review 30 (2015) 45–75.

[16] H. Mousselly-Sergieh, I. Gurevych, Enriching wikidata with frame semantics, in: Proceedings of the 5th Workshop on Automated Knowledge Base Construction, 2016, pp. 29–34.

[17] A. Waagmeester, L. Schriml, A. Su, Wikidata as a linked-data hub for biodiversity data, Biodiversity Information Science and Standards 3 (2019) e35206.

[18] S. Burgstaller-Muehlbacher, A. Waagmeester, E. Mitraka, J. Turner, T. Putman, J. Leong, C. Naik, P. Pavlidis, L. Schriml, B. M. Good, A. I. Su, Wikidata as a semantic framework for the Gene Wiki initiative, Database: The Journal of Biological Databases and Curation 2016 (2016) baw015. doi:10.1093/database/baw015.

[19] G. Faraj, A. Micsik, Enriching wikidata with cultural heritage data from the courage project, in: Research Conference on Metadata and Semantics Research, Springer, 2019, pp. 407–418.

[20] J. Raad, N. Pernelle, F. Saïs, Detection of contextual identity links in a knowledge base, in: Proceedings of the knowledge capture conference, 2017, pp. 1–8.

[21] W. Beek, F. Ilievski, J. Debattista, S. Schlobach, J. Wielemaker, Literally better: Analyzing and improving the quality of literals, Semantic Web 9 (2018) 131–150.

[22] W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, S. Schlobach, Lod laundromat: a uniform way of publishing other people's dirty data, in: International semantic web conference, Springer, 2014, pp. 213–228.

[23] A. Piscopo, L.-A. Kaffee, C. Phethean, E. Simperl, Provenance information in a collaborative knowledge graph: an evaluation of wikidata external references, in: International semantic web conference, Springer, 2017, pp. 542–558.

[24] H. Turki, D. Jemielniak, M. A. H. Taieb, J. E. L. Gayo, M. B. Aouicha, M. Banat, T. Shafee,

E. Prud'Hommeaux, T. Lubiana, D. Das, D. Mietchen, Using logical constraints to validate information in collaborative knowledge graphs: a study of COVID-19 on Wikidata, 2020. doi:10.5281/zenodo.4445363.

[25] M. Färber, F. Bartscherer, C. Menne, A. Rettinger, Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago, Semantic Web 9 (2018) 77–129.

[26] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe, P. Szekely, A study of the quality of wikidata, Journal of Web Semantics (2021).

[27] V. Balaraman, S. Razniewski, W. Nutt, Recoin: relative completeness in wikidata, in: Companion Proceedings of the The Web Conference 2018, 2018, pp. 1787–1792.

[28] J. W. Ratcliff, D. E. Metzener, Pattern matching: The gestalt approach, Dr. Dobb's Journal (1988) 46.

[29] F. Ilievski, D. Garijo, H. Chalupsky, N. T. Divvala, Y. Yao, C. Rogers, R. Li, J. Liu, A. Singh, D. Schwabe, P. Szekely, Kgtk: a toolkit for large knowledge graph manipulation and analysis, in: International Semantic Web Conference, Springer, Cham, 2020, pp. 278–293.

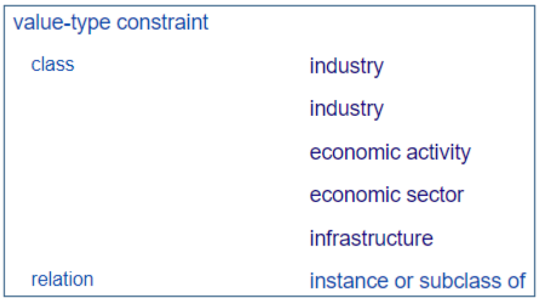
[30] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[31] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, W. Zhang, Knowledge vault: A web-scale approach to probabilistic knowledge fusion, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 601–610.

[32] S. Heindorf, M. Potthast, B. Stein, G. Engels, Vandalism detection in wikidata, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 327–336.

Appendix

A. Value-type Constraint Example



value-type constraint

class	industry
	industry
	economic activity
	economic sector
	infrastructure
relation	instance or subclass of

Figure 2: Example value-type constraint of the property industry (`P452`). The values associated with this property should belong to one of the following types: `[industry, industry, economic activity, economic sector, infrastructure]`, whose respective Qnodes are `[Q8148, Q268592, Q8187769, Q3958441, Q121359]`. The type can be either encoded as an instance-of (`P31`) or a subclass-of (`P279`) property. There are no entities in Wikidata which are exceptions for which this property constraint.

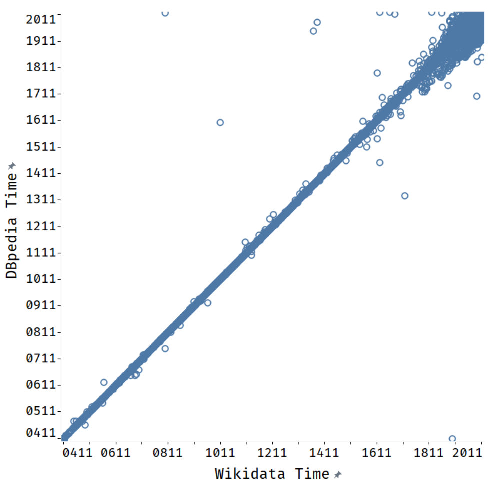
B. Method Implementation

We implement our method using the Knowledge Graph ToolKit (KGTK) [29]. For Getty we obtain paths with maximum length of $L = 4$, for DBpedia $L = 1$. In the schema alignment step, we count property frequency based on a sample up to 200,000 pairs of known subjects and objects (e_w, o_w).

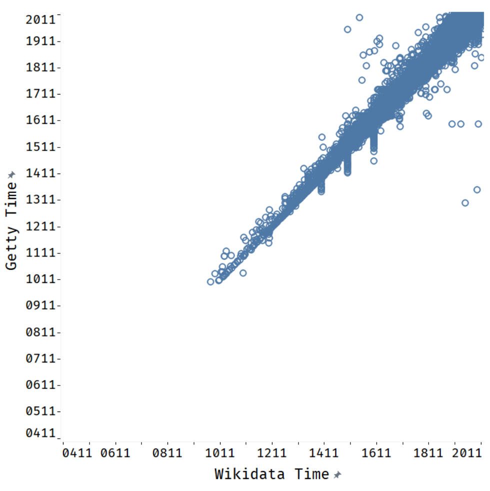
C. Enrichment of Literals

Our method can also be used to enrich literal information about entities. We run our method on two functional properties: `P569` (date of birth) and `P570` (date of death). We compare the obtained results from the external graphs to those found in Wikidata, for entity-property pairs where both Wikidata and the external graph have a value. We compare the results on the finest granularity provided by the the two graphs, which is dates for DBpedia and years to Getty. The results for property `P570` in Figure 3 show a clear trend of the points in scatter plots distributed along the line of $y = x$, which shows high consistency of the date data between the Wikidata and the external KGs.[10] Specifically, we observe that the agreement rate with Wikidata values is 89.28% (1,271,862 out of 1,424,526) for DBpedia and 82.39% for Getty (125,913 out of 152,824). From Getty and DBpedia, our method can enhance Wikidata with novel `P569` values for 35,459 entities and novel `P570` values for 20,664 entities.

[10]We observe a similar trend for `P569`.



(a) DBpedia



(b) Getty

Figure 3: Scatter plots of literal consistency for `P570` (date of death) between Wikidata and the external graph: DBpedia (a) and Getty (b). The plots show the subject-property pairs for which both Wikidata and the external KG have a value. Ticks on both the X- and the Y-axis represent years.