Correcting Assertions and Alignments of Large Scale Knowledge Bases

Jiaoyan Chen [a], Ernesto Jiménez-Ruiz [b,d], Ian Horrocks [a], Xi Chen [c] and Erik Bryhn Myklebust [d,e]
[a] *Department of Computer Science, University of Oxford, Oxford, UK*
E-mails: jiaoyan.chen@cs.ox.ac.uk, ian.horrocks@cs.ox.ac.uk
[b] *City, University of London, London, UK*
E-mail: Ernesto.Jimenez-Ruiz@city.ac.uk
[c] *Jarvis Lab Tencent, Shenzhen, China*
E-mail: ian.horrocks@cs.ox.ac.uk
[d] *Centre for Scalable Data Access (SIRIUS), University of Oslo, Oslo, Norway*
E-mail: jasonxchen@tencent.com
[e] *Norwegian Institute for Water Research, Oslo, Norway*
E-mail: erik.b.myklebust@niva.no

Abstract. Various knowledge bases (KBs) have been constructed via information extraction from encyclopedias, text and tables, as well as alignment of multiple sources. Their usefulness and usability is often limited by quality issues. One common issue is the presence of erroneous assertions and alignments, often caused by lexical or semantic confusion. We study the problem of correcting such assertions and alignments, and present a general correction framework which combines lexical matching, context-aware sub-KB extraction, semantic embedding, soft constraint mining and semantic consistency checking. The framework is evaluated using three representative large scale KBs: DBpedia, an enterprise medical KB and a music KB constructed by aligning Wikidata, Discogs and MusicBrainz.

Keywords: Knowledge Base, Assertion Correction, Alignment Correction, Semantic Embedding, Constraints

1. Introduction

Knowledge Bases (KBs) whose variants are now often known as Knowledge Graphs [21] are playing an increasingly important role in applications such as search engines, question answering, common sense reasoning and data integration. They include general purpose KBs such as Wikidata [59], DBpedia [2] and NELL [37], as well as domain specific KBs such as Discogs and MusicBrainz. Such KBs may be constructed via methods such as extraction from web resources (e.g., DBpedia and NELL) and crowdsourcing (e.g., Wikidata) as well as bespoke knowledge engineering [64], and they often include knowledge from multiple sources that has been integrated via some alignment procedure [6, 66]. Notwithstanding their important role, these KBs still suffer from various quality issues, including constraint violations and erroneous assertions [14, 45], that negatively impact their usefulness and usability. These issues may be due to the knowledge itself (e.g., the core knowledge source of DBpedia, Wikipedia, is estimated to have an error rate of 2.8% [63]), or may be introduced by the knowledge extraction and alignment process.

Existing work on KB quality issues covers not only error detection and assessment, but also quality improvement via completion, canonicalizaiton and so on [45]. Regarding error detection, erroneous assertions can be detected by various methods, including consistency checking with defined, mined or external constraints [27, 47, 55], prediction by machine learning or statistical methods [9, 34, 46], and evaluation by query templates [28]; see Section 2.1 for more details. However, erroneous assertions are typically discarded [8, 41] or reported to curators for manual cor-

rection. Manual curation is expensive and sometimes unfeasible especially when the KB is very large. Actually most KB refinement works focus on completion and error detection, but very few methods or toolkits have been developed to correct the errors [45].

Lertvittayakumjorn et al. [29] and Melo et al. [33] found that most erroneous assertions are due to confusion or lexical similarity leading to entity misuse; for example confusion between *Manchester_United* and *Manchester_City*, two football clubs based in Manchester, UK, can lead to facts about *Manchester_United* being incorrectly asserted about *Manchester_City*. Such errors are common not only in general KBs like DBpedia and Wikidata but also in domain specific KBs like the medical KB used in our evaluation. When aligning multiple KBs by either naive heuristics (e.g., name matching) or state-of-the-art systems (e.g., LogMap [23]), erroneous alignments (i.e., mapping assertions) are also often caused by confusion or lexical similarity. Both [29] and [33] proposed to find an entity to replace either the subject or the object of an erroneous assertion; however, subject replacement used a simple graph metric and keyword matching, which fails to capture the contextual semantics of the assertion, while object replacement relies on Wikipedia disambiguation pages, which may be inaccessible or nonexistent, and again fail to capture contextual semantics.

Other work has focused on quality improvement, for example by canonicalizing *literal assertions* which are those assertions whose objects are literals that represent entities (i.e., entity mentions) [5]; for example, the literal object in the assertion ⟨*Yangtze_River*, *passesArea*, *"three gorges district"*⟩. Replacing this literal with the entity *Three_Gorges_Reservoir_Region* enriches the semantics of the assertion, which can improve query answering. Such literal assertions are pervasive in wiki-based KBs such as DBpedia [2] and Zhishi.me [43], and in open KBs extracted from text; they may also be introduced when two KBs are aligned or when a KB evolves. According to the statistics in [18], DBpedia (ca. 2016) included over 105,000 such assertions using the property *dbp:location* alone. Current methods can predict the type of the entity represented by the literal [18], which is useful for creating a new entity, and can sometimes identify candidate entities in the KB [5], but they do not propose a general correction method; see Section 2.2 for a more details.

In this paper, we propose a general method for correcting *(i)* assertions whose objects are either erroneous entities or literals in an individual KB, and *(ii)* erroneous alignments in a heterogeneous KB as a result of aligning multiple KBs. To this end, we have developed a general framework that exploits related entity estimation, assertion prediction and constraint-based validation, as shown in Figure 2. Given a set of target assertions (i.e., assertions that have been identified as erroneous), it uses semantic relatedness and heuristic rules to identify candidate entities for substitution, extracts a multi-relational graph from the KB (sub-graph) that can model the context of the target assertions, and learns an assertion prediction model using both semantic embeddings and observed features. The model predicts the assertion likelihood for each candidate substitution, and filters out those that lead to unlikely assertions. The framework further verifies the candidate substitutions by checking their consistency w.r.t. property range and cardinality constraints mined from the global KB. The framework finally makes a correction decision, returning a corrected assertion or reporting failure if no likely correction can be identified.

Briefly, this paper makes the following main contributions:

- It proposes a general framework that can correct erroneous entity assertions and literal assertions in an individual KB, and erroneous alignments of multiple KBs;
- For each correction the framework estimates the related entities via various approaches and extracts a context-aware sub-graph, leading to a smaller number of candidates and higher efficiency, thus enabling it to deal with large scale KBs;
- The framework utilizes both semantic embeddings and observed features to capture the local context for correction prediction, and complements the prediction with consistency against "soft" property constraints mined from the global KB;
- It presents an evaluation of the framework with erroneous entity assertions from a medical KB mainly extracted from text, literal assertions from DBpedia, and erroneous alignments in matching artists from Wikidata, Discogs and MusicBrainz.

It is worth noting that for KB quality assurance, detection and correction of erroneous assertions are two critical, challenging and detachable tasks; this study focuses on the correction task. This paper is a substantial extension of a conference paper presented at The Web Conference (WWW) 2020 [4] as it *(i)* adds the

correction of mapping assertions in KB alignment; *(ii)* enhances the techniques to deal with large scale KBs (e.g., extracting a task-specific sub-graph), and evaluates their efficiency and scalability; *(iii)* improves the related entity estimation and assertion prediction for higher robustness; and *(iv)* extends the evaluation with experiments on an additional KB.

The remaining of the paper is organized as follows. Section 2 reviews related work. Section 3 formalises what we mean by a KB and the problem we are addressing. Section 4 presents the technical details of the proposed framework. Section 5 presents an evaluation of the framework. Section 6 concludes with a discussion and some ideas for future work.

2. Related Work

In this section we survey existing work related to assertion validation, which includes erroneous assertion detection, assertion prediction with semantic embeddings and observed features, canonicalization, assertion correction, and repair in KB alignment.

2.1. Assertion Validation

In our work we do not directly address the issue of how to determine the validity of KB assertions, but this is clearly an important consideration, and may suggest directions for correction. One way to identify likely invalid assertions is to check consistency against logical constraints or rules. Explicitly stated KB constraints can be directly used, but these are often weak or even non-existent. Thus, before using the DBpedia ontology to validate assertions, Topper et al. [55] enriched it with class disjointness, and property domain and range costraints, all derived via statistical analysis; Paulheim and Gangemi [47] enriched it via alignment with the DOLCE-Zero foundational ontology. Various constraint and rule languages such as Shapes Constraint Language (SHACL) [27], Rule-Based Web Logics [1] and SPARQL query templates [28], have also been proposed so that external knowledge can be encoded and applied. Similarly, Kharlamov et al. [26] proposed to interpret (a subset of) ontology axioms as (datalog) integrity constraints to assess the completeness of the KB.

As machine learning has developed, various methods have been proposed to encode the semantics of entities and relations into vectors for prediction [61]. The observed features are typically indicators (e.g., paths) extracted for a specific prediction problem. They often work together with other learning and prediction algorithms, including supervised classification (e.g., PaTyBRED [33]), autoencoder (e.g., RDF2Vec [51]), statistical distribution estimation (e.g., SDValidate [46]) and so on. PaTyBRED and SDValidate directly detect erroneous assertions, while RDF2Vec utilizes graph paths to learn intermediate entity representations that can be further used to validate assertions via supervised classification.

In contrast to observed features, which often rely on ad-hoc feature engineering, semantic embeddings (vectors) can be learned by minimizing an overall loss with a score function for modeling the assertion's likelihood [61]. They can be directly used to estimate the assertion likelihood. State-of-the-art methods include those geometric or translation-based models such as TransE [3], TransH [62], TransR [31], and those decomposition or latent factor models such as DistMult [67] and ComplEx [57]. They can also be combined with algorithms such as outlier detection [9] and supervised classification [38] to deal with assertion validation in specific contexts.

On the one hand, the aforementioned methods were mostly developed for KB completion and erroneous assertion detection, and few have been applied in assertion correction, especially the semantic embedding methods. On the other hand, they suffer from various shortcomings that limit their application. Consistency checking depends on domain knowledge of a specific task for constraint and rule definition, while the mined constraints and rules are often weak in modeling local context for disambiguation. Semantic embedding methods are good at modeling contextual semantics in a vector space, but are computationally expensive when learning from large KBs [44] and suffer from low robustness when dealing with real world KBs that are often noisy and sparse [49].

2.2. Canonicalization

Recent work on KB canonicalization is relevant to our related entity estimation. Some of this work focuses on the disambiguation of entity mentions in an open KB extracted from textual data [15, 58, 65]; CESI [58], for example, utilizes side information (e.g., WordNet), semantic embedding and clustering to identify equivalent entity mentions. However, these methods cannot be directly applied in our correction framework as they focus on equality while we aim at estimating relatedness. The contexts are also different as,

unlike entity mentions, literals can only be the object of a triple and usually have fewer links with entities.

Chen et al. [5] and Gunaratna et al. [18] aimed at the canonicalization of literal objects used in assertions with DBpedia object properties (whose objects should be entities). Instead of correcting the literal with an existing entity, they focus on the typing of the entity that the literal represents, which is helpful when a new entity is created for replacement. Although [5] also tried to identify an existing entity to substitute the literal, it suffers from some limitations: the predicted type is used as a constraint for filtering, which is not a robust and general correction method; the related entity estimation is ad-hoc and DBpedia specific; and the type prediction itself only uses entity and property labels, without any other contextual semantics.

2.3. Assertion Correction

We focus on recent studies concerning the automatic correction of erroneous assertions. Some are KB specific. For example, Dimou et al. [11] refined the (relational data to RDF) mappings between Wikipedia data and DBpedia knowledge to correct errors during DBpedia construction, while Pellissier et al. [48] mined correction rules from the edit history of Wikidata to resolve its constraint violations. In contrast, our framework is general and does not assume any additional KB meta information or external data.

Regarding more general approaches, some aim at eliminating constraint violations. For example, Chortis et al. [7, 54] defined and added new properties to avoid violating integrity constraints. These methods ensure KB consistency, but they can neither correct the knowledge itself nor deal with those wrong assertions that satisfy the constraints. Lertvittayakumjorn et al. [29] and Melo et al. [33] both correct assertions by replacing the objects or subjects with correct entities. The former found the substitute by either keyword matching or a simple graph structure metric, while the latter first retrieved candidate substitutes from the Wikipedia disambiguation page (which may not exist, especially for KBs that are not based on Wikipedia) and then ranked them by lexical similarity. Both methods, however, only use simple graph structure or lexical similarity to identify the substitute, and ignore the linkage incompleteness of a KB. In contrast, our method utilizes semantic embeddings to exploit the local context within a sub-graph to predict assertion likelihood, and at the same time uses global property constraints to validate the substitution.

2.4. Alignment Repair

There have been some works for KB alignment repair. Although some ontology alignment systems can also repair instance mappings (e.g., LogMap [23]), most of them focus on concept (class) mappings for expressive ontologies. They usually aim to minimize the undesired logical consequences caused by the mappings (e.g., unsatisfiabilities), by removing some mappings [23, 32, 50, 52, 53, 60] or modifying some axioms of the input ontologies [22] with heuristic, ad-hoc, iterative or interactive methods. The study [8] also removes some identity mappings that lead to constraint violations but focuses on *sameAs* links of Linked Data. Instead of removing some mappings, Euzenat [13] proposed to generate new but less confident mappings after the faulty mappings are discarded. According to the experiment, the solution achieves a high overall precision and more coherent alignments. These repair works aim at maximize logical consistency, but ignore the truth of each mapping. They often adopt some metrics, e.g., the principles of consistency and conservativity proposed by Jimenez-Ruiz et al. [25], to assess the logical consequences. However, in dealing with real world KB alignments, especially those entity alignments with limited logic constraints in the input KBs, erroneous mappings often do not lead to any inconsistency. In contrast, our correction method aims at maximising the "truth" of a mapping and can deal with erroneous mappings that are detected by any method, including logical inconsistency, third party applications or human beings. Moreover, to the best of our knowledge, we are among the first to consider entity mapping repair for large scale KBs.

3. Background

3.1. Knowledge Base

In this study we consider KBs that follow Semantic Web standards including RDF (Resource Description Framework), RDF Schema, OWL (Web Ontology Language)[1] and the SPARQL Query Language [12]. A KB is assumed to be composed of a TBox (terminology) and an ABox (assertions). The TBox usually defines classes (concepts), a class hierarchy (via *rdfs:subClassOf*), properties (roles), and property do-

[1]There is a revision of the Web Ontology Language called OWL 2, for simplicity we also refer to this revision as OWL.

mains and ranges. Each class or property is represented by an Internationalized Resource Identifier (IRI). It may also use a more expressive language such as OWL to express constraints such as class disjointness, cardinality restrictions and so on [17].

The ABox consists of a set of assertions (facts) describing concrete entities (individuals), each of which is represented by an IRI as well. Each assertion is represented by an RDF triple $\langle s, p, o \rangle$, where s is an entity, p is a property and o is either an entity or a literal (i.e., a typed or untyped data value such as a string or integer). s, p and o are known as the subject, predicate and object of the triple. An entity can be an instance of one or more classes. This is expressed by a special triple known as *class assertion* whose predicate is the built-in *rdf:type* property and object is a class. In contrast, the assertion whose p is not a built-in property from the reserved vocabulary of RDF, RDFS or OWL is known as *property assertion*. KB here also includes the case where the TBox is empty (i.e., the KB consists only of an ABox), as is often the case in knowledge graphs. Note the definition of entity in OWL includes class, property and individual. In this paper we use entity to refer to just individual and directly use the terms of class and property. This is often adopted in many knowledge graph works, and makes this paper's presentation more clear.

Such a KB can be accessed by SPARQL queries using a query engine that supports the relevant entailment regime (e.g., RDFS or OWL) [16]; such an engine can, e.g., infer $\langle e_0$ *rdf:type* $c_2 \rangle$, given $\langle e_0$ *rdf:type* $c_1 \rangle$ and $\langle c_1$ *rdfs:subClassOf* $c_2 \rangle$. In addition, large-scale KBs often have a lookup service that enables users to directly access its entities by fuzzy matching; this is usually based on a lexical index that is built with entity labels (phrases defined by *rdfs:label*) and sometimes entity anchor text (short descriptions). DBpedia builds its lookup service[2] using the lexical index of Spotlight [35], while entities of Wikidata can be retrieved, for example, via the backend API of OpenTapioca [10].

3.2. Knowledge Base Alignment

KB alignment refers to the merging of multiple KBs by establishing mappings between classes, properties and instances that represent the same entity [24, 66]. The aligned KBs are regarded as a single KB, where the TBoxes and ABoxes of the input KBs are (respectively) merged; while the original KBs for alignment are named as *component KBs* in this paper. Note that it is possible that multiple entities/concepts/properties from different component KBs have the same IRI, and thus a mapping is not required in these cases.

The mappings are often represented using built-in OWL properties defined by W3C, such as *owl:sameAs*, *owl:equivalentClass* and *owl:equivalentProperty*,[3] or bespoke object properties. They are all referred to as *equivalence properties* in this paper, while their associated assertions are named as *mapping assertions*. These mapping assertions are typically merged into the TBox and/or ABox of the aligned KB. Figure 1 illustrates the alignment of three music KBs that are extracted from Wikidata, Musicbrainz and Discogs,[4] respectively, where the four properties in blue are equivalence properties. For example, the equivalence property *wd:musicbrainzArtist* represents the mappings of artists and artist groups between Wikidata and Musicbrainz; while the assertion $\langle$*wd:Q1299, wd:musicbrainzArtist, musicbrainz:The_Beatles*$\rangle$ states that the subject and the object refer to the same artist group – the English pop-rock band "The Beatles".

As a single KB often has incomplete knowledge, especially for enterprise applications in a specific domain, the alignment of multiple KBs plays an important role in KB enrichment and curation. A typical case is aligning a local KB with domain-specific knowledge and a widely known KB with general knowledge (e.g., aligning the KB from Musicbrainz with the KB from Wikidata). In this case, Musicbrainz has a more comprehensive coverage of music artists, while Wikidata has more wide-ranging knowledge regarding the music artists that it does cover, so the merging can usefully enrich both sources. Following the above example, we will infer $\langle$*musicbrainz:The_Beatles*, *wd:P31, wd:Q5741069*$\rangle$ if $\langle$*wd:Q1299*, *wd:P31, wd:Q5741069*$\rangle$ is declared in the music KB from Wikidata, where *wd:P31* represents the relation "instance of" and *wd:Q5741069* rep-

[2] https://wiki.dbpedia.org/lookup

[3] https://www.w3.org/TR/owl-ref/

[4] Instead of the whole Wikidata but a part of its entities about artists artist groups, and their associated assertions are extracted and used. They are mapped with artists of Musicbrainz KB and Discogs KB by alignment tools and ad-hoc domain knowledge. The latter also include assertions (e.g., those associated with *discogs:hasMember*) that are extracted from their databases. Elements (entities, classes and properties) of the three component KBs are represented with the namespaces of *wd*, *musicbrainz* and *discogs* respectively.

resents the concept "rock band". Moreover, the merging could reveal conflicting information in the two source KBs, which could indicate errors in one or both of the source KBs [21].

3.3. Problem Statement

In this study, we focus on correcting ABox property assertions, i.e., RDF triples $\langle s, p, o \rangle$ where p is not a property from the reserved vocabulary of RDF, RDFS or OWL. As mentioned above, if p is an equivalence property, then we will call the triple a *mapping assertion*; otherwise, if o is an entity, then we will call the triple an *entity assertion*, and if o is a literal then we will call the triple a *literal assertion*. Note that in the case of literal assertions, correction may require more than simple canonicalization; for example, the assertion $\langle$*Sergio_Agüero*, *playsFor*, *"Manchester United"*$\rangle$ should be corrected to $\langle$*Sergio_Agüero*, *playsFor*, *Manchester_City*$\rangle$ instead of just being canonicalized as $\langle$*Sergio_Agüero*, *playsFor*, *Manchester_United*$\rangle$.

We assume that the input is a KB $\mathcal{K}$, and a set $\mathcal{E}$ of assertions that have been identified as incorrect and whose types (literal assertion, entity assertion or mapping assertion) are given. For each literal or entity assertion $\langle s, p, o \rangle$ in $\mathcal{E}$, the proposed correction framework aims at either finding an entity e from $\mathcal{K}$ as an object substitute, such that e is semantically related to o and the new triple $\langle s, p, e \rangle$ is "correct" (i.e., seems to match the true state of affairs in the domain being modelled), or reporting that there is no such entity e in $\mathcal{K}$; while for each mapping assertion $\langle s, p, o \rangle$ in $\mathcal{E}$, the substitute e does not need to be semantically related to o but should be equivalent to s. We focus on correcting the object because most erroneous literal and entity assertions we find in those general KBs from encyclopedias and those industrial domain specific KBs from text and tabular data are caused by the confusion over the objects. Nevertheless, our framework can indirectly correct the subject in the reversed entity assertion, that is, an assertion where subject and object are exchanged and the inverse of the property is used instead. In the case of the erroneous mapping assertions, our framework can already be applied to correct the subject as the equivalence properties in the mapping assertions are symmetric.

We assume the erroneous assertions $\mathcal{E}$ are given. This is reasonable in real world KB curation, as the detection and correction are often regarded as two detached tasks in a pipeline. Erroneous literal assertions can be identified by data type inference and regular expressions as in [18], while erroneous entity assertions and mapping assertions can be detected either manually when the KB is applied in downstream applications or automatically by the methods discussed in Section 2.1. For example, the erroneous assertions of the medical KB are fed back by its industrial deployment in Tencent Technology and users. Specially, erroneous mapping assertions are often fed back by a post processing step involving consistency checking (e.g., as in LogMap [23]) and human interaction [30].

It is important to note that if the KB is an OWL ontology, the set of *object properties* (which connect two entities) and *data properties* (which connect an entity to a literal) should be disjoint. In practice, however, KBs such as DBpedia and some open KBs extracted from the text often do not respect this constraint and some properties are often followed by both entities and literals. The literal assertions for correction are based on such properties (e.g., *playsFor* in the aforementioned erroneous literal assertion example), and it causes no inconsistency to correct them by just replacing their literal objects with entities without changing the properties.

4. Methodology

4.1. Framework

As shown in Figure 2, our assertion correction framework mainly consists of related entity estimation, assertion prediction, constraint-based validation and correction decision making. Related entity estimation identifies those entities that are related to the correct object (substitute) of the assertion. Given a target assertion $t = \langle s, p, o \rangle$, its related entities, ranked by their relatedness, are denoted as RE_t. They are called *candidate substitutes* of the original object o, and the new assertions when o is replaced are called *candidate assertions*. We adopt two techniques — lexical matching and word embedding — to measure relatedness and estimate RE_t. Note that the aim of this step is to ensure high recall; precision is subsequently taken care of via assertion prediction and constraint-based validation over the candidate assertions.

Assertion prediction estimates the likelihood of each candidate assertion. Given a target assertion $t = \langle s, p, o \rangle$ and its related entities RE_t, for each entity e_i in RE_t assertion prediction outputs a score that estimates the likelihood of $\langle s, p, e_i \rangle$. To train such an assertion prediction model, a sub-graph that contains the context

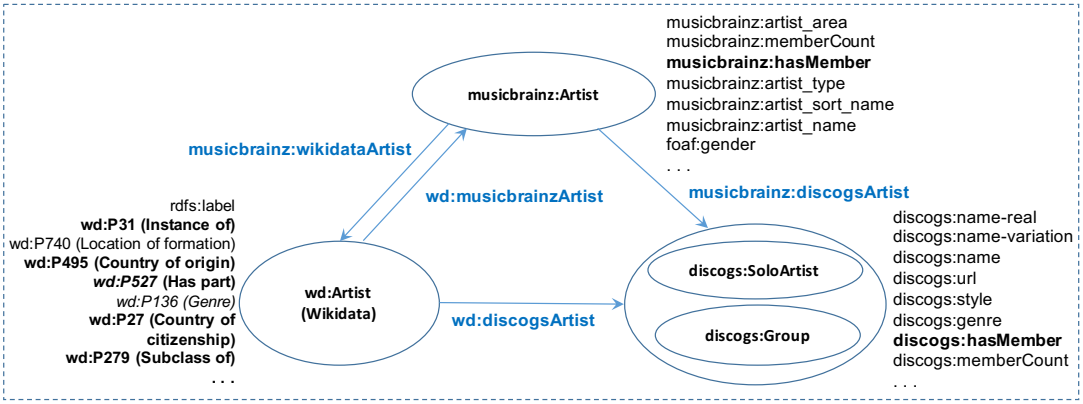

Fig. 1. The Music KB Constructed by Mapping Artists of Wikidata, Discogs and Musicbrainz[5]

of the correction task (i.e., the target assertions $\mathcal{E}$) is first extracted, with the related entities, involved properties and their neighbourhoods; positive and negative assertions are then sampled for training. State-of-the-art semantic embeddings (e.g., TransE [3] and DistMult [67]), as well as some widely used observed features (e.g., path and node) are used to build the assertion prediction model.

Constraint-based validation checks whether a candidate assertion violates constraints on the cardinality or (hierarchical) range of the property, and outputs a consistency score which measures its degree of consistency against such constraints. Such constraints can be effective in filtering out unlikely assertions, but modern KBs such as DBpedia and Wikidata often include only incomplete or weak constraints, or do not respect the given constraints as no global consistency checking is performed. Therefore, we do not assume that there are any cardinality restrictions or range constraints in the KB TBox,[6] but instead use mined constraints, each of which is associated with a supporting degree (probability).

Correction decision making combines the results of related entity estimation, assertion prediction and constraint-based validation; it first integrates the assertion likelihood scores and consistency scores, and then filters out those candidate substitutes that have low scores. Finally, it either reports that no suitable correction was found, or recommends the most likely correction.

[6] Any property range and cardinality constraints that are defined in the TBox, or that come from external knowledge, can be easily and directly injected into the framework.

4.2. Related Entity Estimation

For each target assertion $t = \langle s, p, o \rangle$ in $\mathcal{E}$, related entity estimation first extracts its *anchor phrases* as the input, and then extracts a list RE_t containing up to k most related entities (i.e., $|RE_t| \leqslant k$) by comparing the KB entities against the anchor phrases. The anchor phrases of t are o itself if t is a literal assertion; otherwise, they are the labels and name-like attributes (e.g., *discogs:name-real* in the music KB) of o if t is an entity assertion, and of s if t is a mapping assertion. For a literal/entity assertion t, RE_t is extracted from the whole KB $\mathcal{K}$; while for a mapping assertion t, RE_t is extracted from a component KB of $\mathcal{K}$ where the object o comes from. Our framework supports two lexical matching approaches — traversal based and lexical index (lookup service) based, and one word embedding approach; this allows us to compare the effectiveness of different approaches on different KBs (see Section 5.2).

4.2.1. Lexical Matching (Traversal Based)

For a target assertion t, the lexical matching based approach traverses each entity in the KB (or the component KB of o if t is a mapping assertion) and ranks these entities by their relatedness from high to low. The relatedness between t and an entity is calculated as follows. The lexical matching based approach calculates a string similarity score, i.e., normalized Edit Distance [39], between each anchor phrase of t and each phrase (including labels and name-like attributes) of this entity, and finally adopts the highest score among all the phrase pairs to represent the relatedness of this entity. The score is a value in $[0, 1]$; the higher the score, the more related the entity is. The reason for

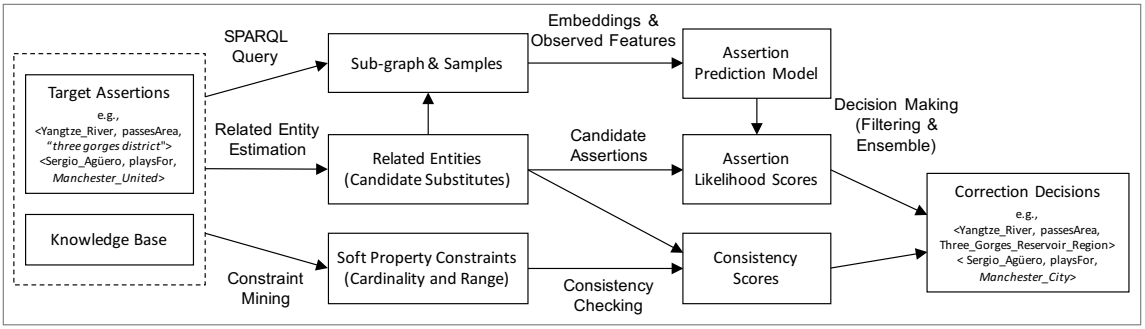

Fig. 2. The Overall Framework for Assertion and Alignment Correction

adopting the highest score is to ensure a high recall of the related entities.

4.2.2. Word Embedding

The general procedure of the word embedding based approach is the same as the above lexical matching based approach, except that it replaces the string similarity score (i.e., normalized Edit Distance) between two phrases by the distance-based similarity score (i.e., cosine similarity) between their embeddings (vectors). To calculate the vector of a phrase, the word embedding based approach *(i)* tokenizes the phrase and removes the stop words, *(ii)* represents each token as a vector using a word embedding model (e.g., *Word2Vec* [36]) that is trained using a large corpus, where tokens out of the model's vocabulary are ignored, *(iii)* calculates the average of the vectors of all the tokens. Compared with lexical matching, word embedding considers the semantics of a word, which assigns a high similarity score to two synonyms. In the above lookup example, "district" becomes noise as it is not included in the label of *dbp:Three_Gorges_Reservoir_Region*, but can still play an important role in the word embedding based approach due to the short word vector distance between "district" and "region". However, in practice, entity misuse is often not caused by semantic confusion, but by similarity of spelling and token composition, where the lexical similarity is high but the semantics may be quite different.

4.2.3. Lookup Service Based

For those KBs with a lexical index, the lexical matching based approach can directly use a lookup service based on the index, which typically returns a list of related entities for each anchor phrase, ranked from higher relatedness (lexical similarity) to lower relatedness, but mostly without a relatedness score; see, e.g., the DBpedia Lookup Service mentioned in Section 3.1. However, anchor phrases derived from (erroneous) entities or literals are often noisy and ambiguous, and direct lookup using such phrases often misses the correct entity. For example, the DBpedia Lookup service returns no entities when given the input "three gorges district", which refers to the entity *dbr:Three_Gorges_Reservoir_Region*. Thus it is common that entities by all the original anchor phrases are less than k. In this case, we retrieve a list of up to k entities by repeating entity lookup using sub-phrases of the longest anchor phrase, starting with the longest sub-phrases and continuing with shorter and shorter sub-phrases until either k entities have been retrieved or all sub-phrases have been used. To extract the sub-phrases, we first tokenize the original phrase, remove the stop words and then concatenate the tokens in their original order for sub-phrases of different lengths.

When the lookup service enables the relatedness score, multiple lists of related entities from different sub-phrases can be merged by re-ordering the entities from high relatedness score to low. When there are no relatedness scores, to merge the entity lists, entities with a front position in their original list are still ranked in the front, while entities with the same position in their original lists are ranked by the following rule: entities by a longer sub-phrase are ranked in the front. Considering two related entity lists $\left[e_1^1, e_2^1, ..., e_k^1\right]$ and $\left[e_1^2, e_2^2, ..., e_k^2\right]$ by two sub-phrases with the first sub-phrase being shorter than the second sub-phrase, the merged related entity list is $\left[e_1^2, e_1^1, e_2^2, e_2^1, ..., e_n^2, e_n^1\right]$.

In comparison with the aforementioned entity traversal with a string similarity score or a word vector similarity score, the lexical index based approach is much more efficient, and makes it easy to utilize multiple items of textual information, such as labels in multiple languages and the abstract (some descriptions of an entity), where different names of an entity are often included. It is worth noting that we can ignore some

anchor phrases in practice according to the real word context; e.g., ignoring anchor phrases of non-English labels if the lexical index is built on English text alone or the word embedding is trained with an English corpus. On the other hand, some more text can sometimes be considered as additional anchor phrases, such as the meaningful entity name in the IRI which often follows the camel case.

4.3. Assertion Prediction

Given related entities RE_t of a target assertion $t = \langle s, p, o \rangle$, assertion prediction is used to estimate a likelihood score for the candidate assertion $\langle s, p, e_i \rangle$, for each entity e_i in RE_t. We train an assertion prediction model, i.e., a classifier, to predict whether a candidate assertion is true or not with a probability. For efficiency in dealing with very large KBs, we first extract a multi-relational sub-graph for the context of the task. We then extract samples with the input of the observed features or semantic embeddings from the sub-graph, and finally train the prediction model.

4.3.1. Sub-graph

Given a KB $\mathcal{K}$ and a set of target assertions $\mathcal{E}$, the sub-graph corresponding to $\mathcal{E}$ is a set of assertions (triples) of $\mathcal{K}$, denoted as $\mathcal{K}_\mathcal{E} = \langle E, P, T \rangle$, where T denotes the assertions, E and P denote the associated entities and properties (relations) respectively. As shown in Algorithm 1, the sub-graph is calculated with three steps: *(i)* extract the seeds — entities and properties involved in the target assertions $\mathcal{E}$, as well as related entities of each assertion in $\mathcal{E}$; *(ii)* extract the neighbourhoods — directly associated assertions of each of the seed properties and entities; *(iii)* re-calculate the properties and entities involved in the target assertions and these neighbourhood assertions as P and E respectively. Note that $\models$ means an assertion is either directly declared in the KB or can be inferred by the KB with entailment reasoning. The statements with $\models$ can be implemented by SPARQL (cf. Section 3.1): Line 13 needs $|P|$ queries each of which retrieves the associated assertions of a given property, while Line 14 needs $2 \times |E|$ queries each of which retrieves the associated assertions of a given subject or object.

4.3.2. Sampling

Both positive and negative samples (assertions) are extracted from the sub-graph $\mathcal{K}_\mathcal{E} = \langle E, P, T \rangle$. The positive samples are composed of two parts: $T_{pos} = T_{sp} \cup T_{pr}$, where T_{sp} refers to assertions whose subjects and properties are among SE (i.e., those subject entities involved in $\mathcal{E}$) and P respectively ($T_{sp} = \{\langle s, p, o \rangle | s \in SE \wedge p \in P \wedge \langle s, p, o \rangle \in T\}$), while T_{pr} refers to those assertions whose objects and properties are among RE (i.e., those related entities involved in $\mathcal{E}$) and P respectively ($T_{pr} = \{\langle s, p, o \rangle | p \in P \wedge o \in RE \wedge \langle s, p, o \rangle \in T\}$). Considering a target assertion set containing $\langle$*Yangtze_River*, *passesArea*, *"three gorges district"*$\rangle$, one potential assertion in T_{sp} is $\langle$*Yangtze_River*, *passesArea*, *Shanghai*$\rangle$, while one assertion in T_{pr} is $\langle$*Yangtze_River*, *hasDam*, *Three_Gorges_Dam*$\rangle$ where we assume *hasDam* is a target property and *Three_Gorges_Dam* is a related entities of *"three gorges district"*. T_{sp} and T_{pr} are calculated by two steps: *(i)* extract all the associated assertions of each property p in P from T; *(ii)* group these assertions according to SE and RE. Compared with an arbitrary assertion in T, the above samples are more relevant to the candidate assertions for prediction. This can help release the domain adaption problem — the data distribution gap between the training and predicting assertions.

The negative samples include two parts as well: $T_{neg} = \tilde{T}_{sp} \cup \tilde{T}_{pr}$, where $\tilde{T}_{sp}$ is constructed according to T_{sp} by replacing the object with a random entity in E, while $\tilde{T}_{pr}$ are constructed according to T_{pr} by re-

Algorithm 1: Sub-graph Extraction ($\mathcal{K}$, $\mathcal{E}$, RE_t)

```
1  Input: (i) The whole KB: K, (ii) The set of target
          assertions: E, (iii) The related entities of each
          target assertion: RE_t, t ∈ E
2  Result: The sub-graph: K_E = ⟨E, P, T⟩
3  begin
4      % Step 1: Extract the seeds
5      SE = {s|⟨s, p, o⟩ ∈ E} % extract subject
         entities
6      P = {p|⟨s, p, o⟩ ∈ E} % extract target
         properties
7      RE = ∪_{t∈E} RE_t % The union of related entities
8      E = SE ∪ RE
9      foreach ⟨s, p, o⟩ in E do
10         if o is an entity then
11             E = E ∪ {o} % add object entity
12     % Step 2: Extract the neighbourhoods
13     T = {⟨s, p, o⟩|p ∈ P ∧ K ⊨ ⟨s, p, o⟩}
14     T =
         T ∪ {⟨s, p, o⟩|o ∈ E ∧ s ∈ E ∧ K ⊨ ⟨s, p, o⟩}
15     % Step 3: Re-calculate entities and properties
16     E = E ∪ {s|⟨s, p, o⟩ ∈ T} ∪ {o|⟨s, p, o⟩ ∈ T}
17     P = P ∪ {p|⟨s, p, o⟩ ∈ T}
18     return K_E = ⟨E, P, T⟩
```

placing the subject with a random entity in E. Take T_{sp} as an example, for each of its assertion $\langle s, p, o\rangle$, an entity $\tilde{e}$ is randomly selected from E for a synthetic assertion $\tilde{t} = \langle s, p, \tilde{e}\rangle$ such that $\mathcal{K} \nvDash \tilde{t}$, where $\nvDash$ denotes that an assertion is neither declared nor can be inferred, and $\tilde{t}$ is added to $\tilde{T}_{sp}$. Considering the above positive assertion $\langle$*Yangtze_River*, *passesArea*, *Shanghai*$\rangle$, one example of its conrresponding negative assertion is $\langle$*Yangtze_River*, *passesArea*, *Three_Gorges_Dam*$\rangle$. In implementation, we get $\mathcal{K} \nvDash \tilde{t}$ if $\tilde{t} \notin T$, as T is extracted from the KB with inference. T_{pr} is constructed similarly. The size of T_{pos} and T_{neg} is balanced.

4.3.3. Observed Features

We extract three kinds of observed features as the prediction model's input — *(i)* the path feature which represents potential relations that can connect the subject and object, *(ii)* the node feature which represents the likelihood of the subject being the head of the property and the likelihood of the object being the tail of the property, and *(iii)* the name similarity which represents degree of equality of two entities w.r.t. their names. Note that the name similarity is used for correcting mapping assertions.

For the path feature, we limit the path depth to two, to reduce computation time and feature size, since both have exponential complexity w.r.t. the depth. In fact, it has been shown that paths of depth one are already quite effective. They outperform the state-of-the-art KB embedding methods such as DistMult and TransE, together with the node feature on some benchmarks [56]. Meanwhile the predictive information of a path will vanish as its depth increases. In calculation, we first extract paths of depth one: FP^1_{so} and FP^1_{os}, where FP^1_{so} represents properties from s to o (i.e., $\{p_0|\langle s, p_0, o\rangle \in T\}$), while FP^1_{os} represents properties from o to s (i.e., $\{p_0|\langle o, p_0, s\rangle \in T\}$). Next we calculate paths of depth two (ordered property pairs) in two directions as well: $FP^2_{so} = \{(p_1, p_2)|\langle s, p_1, e\rangle \in T \wedge \langle e, p_2, o\rangle \in T\}$, $FP^2_{os} = \{(p_1, p_2)|\langle o, p_1, e\rangle \in T \wedge \langle e, p_2, s\rangle \in T\}$. Finally, we merge these paths: $FP = FP^1_{so} \cup FP^1_{os} \cup FP^2_{so} \cup FP^2_{os}$. The paths of an assertion (sample) are encoded into a feature vector, denoted as f^p, in the following way. We collect and order all the unique paths from the training assertions as a candidate set, set the dimension of the feature vector to the number of these paths, and let one path correspond to one slot of the feature vector. When an assertion's FP is encoded into f^p, a slot of the vector is set to 1 if the slot's corresponding path is in FP and 0 otherwise.

The node feature includes two binary variables: $f^n = [v_s, v_o]$, where v_s denotes the likelihood of the subject while v_o denotes the likelihood of the object. Namely, $v_s = 1$ if there exists some entity o' such that $\langle s, p, o'\rangle \in T$ and $v_s = 0$ otherwise. $v_o = 1$ if there exists some entity s' such that $\langle s', p, o\rangle \in T$ and $v_o = 0$ otherwise. The name similarity feature, denoted as f^s is the string similarity score (i.e., normalized Edit Distance) of the names of two entity. As in related entity estimation, entity labels of different languages as well as name alike attributes are considered as the name, and the highest score among all the phrase pairs of two entities is adopted.

Finally we calculate f^p, f^n and f^s, and concatenate them for each sample in $T_{pos} \cup T_{neg}$, and train an assertion prediction model with a basic supervised classifier named Multiple Layer Perception (MLP):

$$\mathcal{M}_{pns} \xleftarrow[T_{pos} \cup T_{neg}]{\text{classifier e.g., MLP}} [f^p, f^n, f^s]. \tag{1}$$

Note we can also use f^p, f^n and f^s independently or with an arbitrary combination for training. The trained model is denoted by $\mathcal{M}$ with the corresponding subscripts.

We also adopt the path-based latent feature learned by the state-of-the-art algorithm RDF2Vec [51], as a baseline. RDF2Vec first extracts potential outstretched paths of an entity by e.g., graph walks, and then learns embeddings of the entities through the neural language model *Word2Vec*. In training, we encode the subject and object of an assertion by their RDF2Vec embeddings, encode its property by a one-hot vector (each slot of the vector corresponds to one property; the slot of a specified property is set to 1 while the remaining are set to 0), concatenate the three vectors, and use the same classifier MLP. The trained model is denoted as $\mathcal{M}_{r2v}$.

4.3.4. Semantic Embeddings

A number of semantic embedding algorithms have been proposed to learn the vector representation of KB properties and entities. One common way is to define a scoring function to model the truth of an assertion, and use an overall loss for learning the semantics of a KB. We adopt and evaluate five state-of-the-art algorithms — TransE [3], TransH [62] and TransR [31], DistMult [67] and ComplEx [57]. The former three are translation-based models, while the latter two are latent factor models. For high efficiency in dealing with very large KBs, we learn the embeddings from the relevant sub-graph as described in Section 4.3.1.

TransE tries to learn a vector representation space such that o is a nearest neighbor of $s + p$ if an assertion $\langle s, p, o\rangle$ holds, and o is far away from $s + p$ otherwise. $+$ denotes the vector add operation. To this end, the score function of $t = \langle s, p, o\rangle$, denoted as $g(t)$, is defined as $d(\boldsymbol{e}_s + \boldsymbol{e}_p, \boldsymbol{e}_o)$, where d is a dissimilarity (distance) measure such as L_2 norm, while $\boldsymbol{e}_s$, $\boldsymbol{e}_p$ and $\boldsymbol{e}_o$ are embeddings of s, p and o respectively. The embeddings have the same dimension, which can be configured, and are initialized by a one-hot encoding. In learning, a gradient descent optimization algorithm is used to minimize the following margin-based ranking loss:

$$L = \sum_{t\in T, t\to\tilde{t}} [\gamma + g(t) - g(\tilde{t})]_+ \tag{2}$$

where $\gamma > 0$ is a hyper parameter, $[\cdot]_+$ denotes extracting the positive part, and $\tilde{t}$ represents a negative assertion of t, generated by randomly replacing the subject or object with an entity in E. TransH and TransR are extensions of TransE. TransH first projects the subject vector and the object vector to the space of the relation vector, and then calculate the score of a triple. Namely, $g(t) = d(\boldsymbol{e}'_s + \boldsymbol{e}_p, \boldsymbol{e}'_o)$, where $\boldsymbol{e}'_s = \boldsymbol{e}_s - w_p^T \boldsymbol{e}_s w_p$ and $\boldsymbol{e}'_o = \boldsymbol{e}_o - w_p^T \boldsymbol{e}_o w_p$ denote the projections, w_p denotes the vector of projection weights to learn. TransR transforms the subject vector and the object vector into the space of the relation vector by multiplying them with a matrix. Namely, $g(t) = d(\boldsymbol{e}'_s + \boldsymbol{e}_p, \boldsymbol{e}'_o)$, where $\boldsymbol{e}'_s = M_p \boldsymbol{e}_s$ and $\boldsymbol{e}'_o = M_p \boldsymbol{e}_o$, M_p denotes the project matrix to learn.

DistMult is a special form of the bilinear model which represents a property (relation) by a matrix. DistMult assumes that the non-diagonal entries in the relation matrices are zero. The score function of an assertion is defined as $g(t) = \boldsymbol{e}_s^T diag(\boldsymbol{e}_p)\boldsymbol{e}_o$, where $diag(\cdot)$ denotes the matrix's diagonal elements (a vector). As TransE, the entity embeddings and property embeddings are initialized by one-hot encoding, with a configurable dimension. A similar margin-based ranking loss as Equation (2) is used for training with a gradient descent optimization algorithm. ComplEx extends DistMult by introducing complex-valued embeddings so as to better model asymmetric relations. Its score function of an assertion is defined as $g(t) = Re(\boldsymbol{e}_s^T diag(\boldsymbol{e}_p)\bar{\boldsymbol{e}_o})$, where $\bar{\boldsymbol{e}_o}$ denotes the complex conjugate of $\boldsymbol{e}_o$ and $Re(\cdot)$ means taking the real part of a complex value.

In prediction, the score of an assertion $g(t)$ can be calculated with the corresponding scoring function and the embeddings of its subject, property and object. The lower the score, the more likely the assertion. To make it consistent with the predicted score by MLP and observed features, which is in $[0, 1]$ and proportional to the assertion likelihood, we calculate the assertion likelihood score as

$$y_{score} = \frac{1}{1 + exp(-\frac{1}{g(t)})}, \tag{3}$$

where exp denotes the exponential function. We denote the assertion prediction model by TransE, TransH, TransR, DistMult and ComplEx as $\mathcal{M}_{te}$, $\mathcal{M}_{th}$, $\mathcal{M}_{tr}$, $\mathcal{M}_{dm}$ and $\mathcal{M}_{ce}$ respectively.

4.4. Constraint-based Validation

We first mine two kinds of soft constraints — property cardinality and hierarchical property range from the KB, and then use a consistency checking algorithm to validate those candidate assertions.

4.4.1. Property Cardinality

Given a property p, its soft cardinality is represented by a probability distribution $D^p_{car}(k) \in [0, 1]$, where $k \geqslant 1$ is an integer that denotes the cardinality. It is calculated as follows: *(i)* get all the property assertions whose property is p, denoted as $T(p)$, and all the involved subjects, denoted as $S(p)$, *(ii)* count the number of the object entities associated with each subject s in $S(p)$ and p: $ON(s, p) = |\{o|\langle s, p, o\rangle \in T(p)\}|$, *(iii)* find out the maximum object number: $ON^p_{max} = max\{ON(s, p)|s \in S(p)\}$, and *(iv)* calculate the property cardinality distribution as:

$$D^p_{car}(k) = \frac{|\{s \in S(p)|ON(s, p) = k\}|}{|S(p)|} \quad \text{with } k = 1, ..., ON^p_{max}, \tag{4}$$

where $|\cdot|$ denotes the size of a set. Specially $ON^p_{max} = 0$ if $T(p)$ is empty. $D^p_{car}(k > n)$ is short form of $\sum_{i=n+1}^{ON^p_{max}} D^p_{car}(k = i)$, denoting the probability that the cardinality is larger than n. In implementation, $T(p)$ can be accessed by one time SPARQL query, while the remaining computation has linear time complexity w.r.t. $|T(p)|$.

The probability of cardinality k is equal to the ratio of the subjects that are associated with k different entity objects. For example, considering a property *hasParent* that is associated with 10 different subjects (persons) in the KB, if one of them has one object

(parent) and the remaining have two objects (parents), then the cardinality distribution is: $D_{car}(k=1) = 1/10$ and $D_{car}(k=2) = 9/10$. Note that although such constraints follow Closed Word Assumption and Unique Name Assumption, they are suitable for our method. On the one hand, probabilities are estimated to represent the supporting degree of a constraint by the ABox. One the other hand, they are used in an approximate model to validate candidate assertions instead of as new and totally true knowledge for KB TBox extension.

4.4.2. Hierarchical Property Range

Given a property p, its range constraint consists of *(i) specific range* which includes the most specific classes of its associated objects, denoted as C^p_{sp}, and *(ii) general range* which includes ancestors of these most specific classes, denoted as C^p_{ge}, with top classes such as *owl:Thing* being excluded. A most specific class of an entity refers to one of the most fine grained classes. The most specific classes of an entity refers to the most fine grained classes that the entity is an *instance of* according to the class assertions and class hierarchy in the KB. Namely, given an entity e, a class c is one specific class of e if $\langle e,\ rdf{:}type,\ c\rangle$ and there exists no class c' that satisfies $c' \neq c$, $\langle e,\ rdf{:}type,\ c'\rangle$ and $\langle c',\ rdfs{:}subClassOf,\ c\rangle$. Note that there could be multiple such classes as the entity could be asserted to be an instance of multiple classes belonging to independent branches of the hierarchy. General classes of an entity are those that subsume one or more of the specific classes in the KB via *rdfs:subClassOf* assertions.

Each range class c in C^p_{sp} (C^p_{ge} resp.) has a probability in $[0,1]$ that represents its supporting degree by the KB, denoted as $D^p_{sp}(c)$ ($D^p_{ge}(c)$ resp.). C^p_{sp}, C^p_{ge} and the supporting degrees are calculated by the following steps: *(i)* get all the object entities that are associated with p, denoted as $OE(p)$; *(ii)* infer the specific and general classes of each entity oe in $OE(p)$, denoted as $C_{sp}(p, oe)$ and $C_{ge}(p, oe)$ respectively, and at the same time collect C^p_{sp} as $\cup_{oe\in OE(p)} C_{sp}(p, oe)$ and C^p_{ge} as $\cup_{oe\in OE(p)} C_{ge}(p, oe)$; *(iii)* compute the supporting degrees:

$$\begin{cases} D^p_{sp}(c) = \frac{|\{oe|oe\in OE(p), c\in C_{sp}(p,oe)\}|}{|OE(p)|}, c \in C^p_{sp}, \\ D^p_{ge}(c) = \frac{|\{oe|oe\in OE(p), c\in C_{ge}(p,oe)\}|}{|OE(p)|}, c \in C^p_{ge}. \end{cases} \tag{5}$$

The degree of each range class is the ratio of the objects that are instances of the class, as either directly declared in the ABox or inferred by *rdfs:subClassOf*. The implementation needs one time SPARQL query to get $OE(p)$, and $|OE(p)|$ times SPARQL queries to get the specific and ancestor classes. The remaining computation has linear time complexity w.r.t. $|OE(p)|$. As property cardinality, the range is also used for approximating the likelihood of candidate assertions, with a consistency checking algorithm introduced bellow.

4.4.3. Consistency Checking

As shown in Algorithm 2, constraint checking acts as a model to estimate the consistency of an assertion against soft constraints of hierarchical property range and cardinality. Given a candidate assertion $t = \langle s, p, e\rangle$, the algorithm first checks the property cardinality, with a parameter named maximum cardinality exceeding rate $\sigma \in (0, 1]$. Line 5 counts the number of entity objects that are associated with s and p in the KB, assuming that the correction is made (i.e., t has been added into the KB). Note that $1 \leqslant n \leqslant ON^p_{max}+1$. Line 6 calculates its exceeding rate r w.r.t. ON^p_{max}, where $r \in (-\infty, 1]$. In Line 8, $ON^p_{max} = 0$ indicates that p is highly likely to be used as a data property in the KB. This is common in correcting literal assertions: one example is the property *hasName* whose objects are phrases of entity mentions but should not be replaced by entities. In this case, it is more reasonable to report that the object substitute does not exist, and thus the algorithm sets the cardinality score y_{car} to 0.

Another condition of setting y_{car} to 0 is $r \geqslant \sigma$. Specially, when σ is set to 1.0, $r \geqslant \sigma$ (i.e., $ON^p_{max} = 1$, $n = 2$) means that p is a object property with functionality in the KB but the correction violates this constraint. Note that n can exceed ON^p_{max} by a small degree which happens when ON^p_{max} is large. For example, when σ is set to 0.5, $r = 0.25$ (i.e., $ON^p_{max} = 4$ and $n = 5$) is allowed. Line 11 to 16 calculate the property cardinality score y_{car} as the probability of being a functional property ($n = 1$), or as the probability of being a none-functional property ($n > 1$). Specially, we punish the score when $n > ON^p_{max}$ (i.e., $r > 0$) by multiplying it with a degrading factor $1 - r$: the higher exceeding rate, the more it degrades.

Line 17 to 21 calculate the property range score y_{ran}, by combining the specific range score $y_{ran,c}$ and the general range score $y_{ran,g}$ with their importance weights ω_c and ω_g. Usually we make the specific range more important by setting ω_c and ω_g to e.g., 0.8 and 0.2 respectively. Line 19 computes $y_{ran,c}$ and

$y_{ran,g}$: the score is higher if more classes of the objects are among the range classes, and these classes have higher range degrees. For example, considering the property *bornIn* with the following range cardinality: $C^p_{sp} = \{City, Town, Place\}$, $C^p_{ge} = \{Place\}$, $D^p_{sp}(City) = 0.5$, $D^p_{sp}(Town) = 0.4$, $D^p_{sp}(Place) = 0.05$ and $D^p_{ge}(Place) = 0.95$, we will have *(i)* $y_{ran,c} = 1-(1-0.5)(1-0.05) = 0.525$ and $y_{ran,g} = 0.95$ if $C(e) = \{City, Place\}$, *(ii)* $y_{ran,c} = 0.05$ and $y_{ran,g} = 0.95$ if $C(e) = \{Village, Place\}$, and *(iii)* $y_{ran,c} = 0$ and $y_{ran,g} = 0$ if $C(e) = \{Professor, Person\}$. The order of the consistency degree against the property range is: $\{City, Place\} > \{Village, Place\} > \{Professor, Person\}$.

The algorithm finally returns the property cardinality score y_{car} and the property range score y_{ran}. The former model is denoted as $\mathcal{M}_{car}$ while the latter is denoted as $\mathcal{M}_{ran}$. According to some empirical analysis, we can multiply or average the two scores, as the final model of consistency checking, denoted as $\mathcal{M}_{car+ran}$.

4.5. Correction Decision Making

Given a target assertion t in $\mathcal{E}$, and its top-k related entities RE_t, for each entity e_i in RE_t, the correction framework *(i)* calculates the assertion likelihood score y^l_i with an assertion prediction model ($\mathcal{M}_{pns}$, $\mathcal{M}_{te}$, $\mathcal{M}_{th}$, $\mathcal{M}_{tr}$, $\mathcal{M}_{dm}$ or $\mathcal{M}_{ce}$), and the consistency score y^c_i with $\mathcal{M}_{car}$, $\mathcal{M}_{ran}$ or $\mathcal{M}_{car+ran}$; *(ii)* separately normalizes y^l_i and y^c_i into $[0,1]$ according to all the predictions by the corresponding model for $\mathcal{E}$; *(iii)* ensembles the two scores by simple averaging: $y_i = \frac{(y^l_i + y^c_i)}{2}$; *(iv)* filters out e_i from RE_t if $y_i < \tau$. Note e_i is always kept if t is a literal assertion and its literal is exactly equal to the label of e_i. The related entities after filtering keep their original order in RE_t, and are denoted as RE'_t. τ is a parameter in $[0,1]$ that needs to be adjusted with a developing data set. In the end, the decision procedure decides that there is no entity in the KB that can replace the object of t, if RE'_t is empty, or returns the top-1 entity in RE'_t as the object substitute if RE'_t is NOT empty. The ensemble of the assertion prediction score and constraint-based validation score is not a must. Either of them can make a positive impact independently, while their ensemble can make the performance higher in most cases, as evaluated in Section 5.4.

Algorithm 2: Consistency Checking ($\mathcal{M}_{ran}$, $\mathcal{M}_{car}$)

Input: *(i)* A candidate assertion: $t = \langle s, p, e\rangle$, *(ii)* property cardinality constraint: $\langle D^p_{car}, ON^p_{max}\rangle$, *(iii)* the maximum cardinality exceeding rate: $\sigma \in (0,1]$, *(iv)* hierarchical property range constraint: $\langle D^p_{sp}, D^p_{ge}, C^p_{sp}, C^p_{ge}\rangle$, *(v)* weights of the specific range and general range: $\langle \omega_c, \omega_g\rangle$

Result: y_{car}: score that t is consistent with the property cardinality; y_{ran}: score that t is consistent with the property range

begin

  % *count the number of object entities*

  $n = |\{o | \mathcal{K} \models \langle s, p, o\rangle, o \text{ is entity}\} \cup \{e\}|$;

  $r = \frac{(n - ON^p_{max})}{ON^p_{max}}$; % *calculate the exceeding rate*

  % *no object entities are associated with p in the KB, or the cardinality exceeds the maximum by a specific rate*

  if $ON^p_{max} = 0 \parallel r \geqslant \sigma$ **then**

    $y_{car} = 0$;

  else

    if $n = 1$ **then**

      % *probability as a functional property*

      $y_{car} = D^p_{car}(k = 1)$;

    else

      % *probability as a none-functional property*

      $y_{car} = \begin{cases} D^p_{car}(k > 1), & \text{if } r \leqslant 0 \\ (1 - r) \cdot D^p_{car}(k > 1), & \text{else} \end{cases}$

  $C(e) = \{c | \mathcal{K} \models \langle e, rdf{:}type, c\rangle\}$; % *get the object's classes*

  % *calculate the constraint score of specific and general ranges*

  $\begin{cases} y_{ran,c} = 1 - \prod_{c \in C^p_{sp} \cap C(e)} (1 - D^p_{ran}(c)), \\ y_{ran,g} = 1 - \prod_{c \in C^p_{ge} \cap C(e)} (1 - D^p_{ran}(c)); \end{cases}$

  % *calculate the overall range constraint score*

  $y_{ran} = \omega_c \cdot y_{ran,c} + \omega_g \cdot y_{ran,g}$;

  return y_{car}, y_{ran}

5. Evaluation

5.1. Experiment Settings

5.1.1. Data

In our experiment, we correct *(i)* literal assertions in DBpedia [2], *(ii)* entity assertion in an enterprise medical KB whose TBox is defined by clinic experts and ABox is extracted from medical articles

(text) by some open information extraction tools (cf. more details in [42]), and *(iii)* mapping assertions in a music KB that is constructed by aligning artist and artist group entities in Wikidata and two KBs transformed from Musicbrainz and Discogs databases (cf. Figure 1). The data are representative of three common situations: errors of DBpedia are mostly inherited from the source; errors of the medical KB are mostly introduced by the extraction procedure; while errors of the music KB are caused by KB alignment. DBpedia is accessed via its official Lookup service, SPARQL Endpoint[7] and entity label dump (for related entity estimation with *Word2Vec*). The medical KB contains knowledge about disease, medicine, treatment, symptoms, foods and so on, with around 800 thousand entities, 7 properties, 48 classes, 4 million property assertions. The music KB includes around 89.9 thousand, 1.5 million and 6.4 million entities of artist and artist group from Wikidata, Musicbrainz and Discogs, respectively; while its equivalence properties *wd:musicbrainzArtist*, *wd:discogsArtist*, *musicbrainz:wikidataArtis* and *musicbrainz:discogsArtist* have around 50.9 thousand, 45.3 thousand, 178.3 thousand and 590.1 thousand assertions associated, respectively.

Regarding DBpedia, we reuse the real world literal assertions proposed by [5, 18]. As our task is not typing the literal, but replacing it by an entity, literals containing multiple entity mentions are removed, while properties with insufficient literal objects are complemented with more literals from DBpedia. We annotate each assertion with a ground truth (GT), which is either a correct replacement entity from DBpedia (i.e., *Entity GT*) or none (i.e., *Empty GT*). Ground truths are carefully checked using DBpedia, Wikipedia, and multiple external resources. Regarding the medical KB and the music KB, we use a set of assertions with erroneous entity objects that have been discovered and collected during deployment of the KB in enterprise products in Tencent Technology and Oxford Semantic Technology, respectively. For the music KB, we select two equivalence properties: *wd:musicbrainzArtist* which aligns a Wikidata entity to a Musicbrainz entity, and *wd:discogsArtist* which aligns a Wikidata entity to a Discogs entity. The GT annotations of the medical KB erroneous assertions have been added with the help of clinical experts, while those of the music KB erroneous mapping assertions have been added by checking the links between the web pages of artists and artist groups (e.g., the link to a Wikidata entity on the Musicbrainz page of an artist), and detailed information e.g., an artist's biography and album. For convenience, we call the above three target assertion sets DBP-Lit, MED-Ent and MUS-Map respectively.[8] Some statistics are shown in Table 1. Each target assertion set is randomly splitted into a validation set (10%) for developing the framework (e.g., hyper parameter adjustion) and a testing set (90%) for measuring the performance. Note that no training set is needed since the framework needs no annotated assertions (the samples for training the assertion prediction model are extracted from the existing assertions of the KB itself).

5.1.2. Settings

In the evaluation, we first analyze related entity estimation (Section 5.2) and assertion prediction (Section 5.3) independently. For related entity estimation, we report the recall of Entity GTs of different methods with varying top-*k* values, based on which a suitable method and a suitable *k* value can be selected for the framework. A higher recall with a lower *k* value means a better approach. For assertion prediction, we compare the performance of different semantic embeddings and observed features, and at the same time analyze the impact of extracting the sub-KB on both accuracy and efficiency, using those target assertions whose Entity GTs are recalled in related entity estimation. The related entities of a target assertion are first ranked according to the predicted score, and then standard metrics including Hits@1, Hits@5 and MRR (Mean Reciprocal Rank)[9] are calculated. Hits@1 (Hits@5 resp.) is the recall of the GT by the top 1 (5 resp.) entities of the rank, while MRR indicates the position of the GT in the rank. Values of all these metrics are among $[0, 1]$; a higher value indicates a higher performance.

Next we evaluate the overall results of the assertion correction framework (Section 5.4), where several baselines (the original DBpedia lookup, DBpedia lookup with sub-phrases, matching with Edit Distance and *word2vec*) are compared with, and the impact of assertion prediction and constraint-based validation is analyzed. Three metrics are adopted: *(i) Correction Rate* which is the ratio of the target assertions that are corrected with right substitutes, among all the target

[7] http://dbpedia.org/sparql

[8] DBP-Lit data and its experiment codes: https://github.com/ChenJiaoyan/KG_Curation

[9] https://en.wikipedia.org/wiki/Mean_reciprocal_rank

	Target Assertions #	Target Assertions with Entity GT #	Properties #	Subjects #
DBP-Lit	725	499	127	668
MED-Ent	272	225	7	200
MUS-Map	200	195	2	200

Table 1
Statistics of DBP-Lit, MED-Ent and MUS-Map.

assertions with Entity GTs; *(ii) Empty Rate* which is the ratio of the target assertions that are corrected with none, among all the target assertions with Empty GTs; *(iii) Accuracy* which is the ratio of the truly corrected target assertions by either substitutes or none, among all the target assertions. Note that accuracy is an overall metric considering both correction rate and empty rate. Either high (low resp.) correction rate or empty rate can lead to high (low resp.) accuracy. With the overall results, we finally analyze the constraint-based validation with more details. Due to the very simple range and cardinality of the two target mapping properties, filtering by constraint-based validation is not applied in MSU-Map.

The reported results are based on the following setting (unless otherwise specified). In related entity estimation, *Word2Vec* [36] trained using the Wikipedia article dump in June 2018 is used for word embedding. In assertion prediction, the hidden layer size of MLP is set to 150; the embedding size of both entities and properties is set to 100; TransE, TransH and TransR are trained by the SGD optimizer, while DistMult and ComplEx are trained by the Adagrad optimizer; other hyper parameters such as the number of epochs and the margin hyper parameter γ are set such that the highest MRR are achieved on the validation set of assertions. Regarding the baseline RDF2Vec, pre-trained versions of DBpedia entities with different settings by Mannheim University[10] are tested, and the results with the best MRR are reported. In constraint-based validation, σ, ω_c and ω_g are set to 1.0, 0.8 and 0.2 respectively, according to the algorithm insight. Some other reasonable settings explored can also achieve similar results. The embeddings are trained by GeForce GTX 1080 Ti with the implementation of OpenKE [19], while the remaining is computed by Intel(R) Xeon(R) CPU E5-2670 @2.60GHz and 32G RAM.

5.2. Related Entity Estimation

We compare different methods and settings used in related entity estimation. Figure 3 shows the obtained recall w.r.t. Entity GTs by top-k related entities. Lookup* denotes our lookup solution with sub-phrases. Edit Distance and *Word2Vec* are based on the entity labels alone for DBP-Lit and MED-Ent, but adopt both entity labels and name alike attributes (including *discogs:name*, *discogs:name-real* and *discogs:name-variation* of Discogs entities, *musicbrainz:artist_name* and *musicbrainz:artist_sort_name* of Musicbrainz entities) for MUS-Map.

First, we find that the lexical matching based methods (Lookup, Lookup* and Edit Distance) have much higher recall than *Word2Vec*, on DBP-Lit and MED-Ent. The reason for DBP-Lit may lie in the Lookup service provided by DBpedia, which takes not only the entity label but also the abstract (i.e., some textual descriptions) of an entity into consideration. The latter provides more semantics, some of which, such as different names and background description, is very helpful for recalling the right entity. The reason for MED-Ent, according to some empirical analysis, is that the erroneous objects are often caused by lexical confusion, such as misspelling and misusing of an entity with similar tokens. On MUS-Map, Edit Distance also has higher recall than *Word2Vec* as many words of the artist name are either out of the vocabulary of the *Word2Vec* or have no specific meaning. Second, our lookup solution with sub-phrases, i.e., Lookup*, as expected, outperforms the original Lookup. For example, when both curves are stable, their recalls are around 0.88 and 0.81, respectively,

The target of related entity estimation in our framework is to have a high recall with a k value that is not too large (so as to avoid additional noise and limit the size of the sub-graph for efficiency). In real application, the method and k value can be set by analyzing the recall curve of related entities. According to the recall curves which increase dramatically at the beginning and then keep stable, our framework uses Lookup* with $k = 30$ for DBP-Lit, Edit Distance with $k = 76$ for MED-Ent, and Edit Distance with $k = 40$ for MUS-Map.

[10] https://bit.ly/2M4TQOg

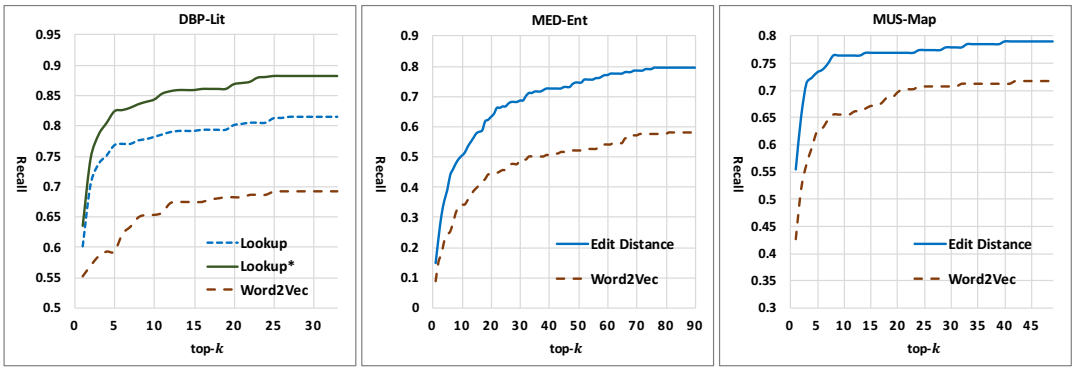

Fig. 3. The recall w.r.t. Entity GTs by top-k related entities

5.3. Assertion Prediction

5.3.1. Impact of Models

The results of different assertion prediction methods are shown in Table 2, where the sub-graph is used for training. The baseline Random means randomly ranking the related entities, while AttBiRNN refers to the attentive bidirectional Recurrent Neural Networks that classify the word sequence composed of the labels of the subject, property and object. The sequence size is fixed, and *Word2Vec* is used to encode its words. It was used in [5] for literal object typing, with good performance achieved. First of all, the results verify that either latent semantic embeddings or observed features with Multiple Layer Perception are effective for all the three benchmarks: MRR, Hits@1 and Hits@5 are all dramatically improved in comparison with Random. AttBiRNN performs poorly on both DBP-Lit and MED-Ent but well on MUS-Map (better than the semantic embeddings). The is because entity names are often effective in judging the entity alignment; while AttBiRNN is good at predicting the semantic equality of the two sub-sequences.

We also find that concatenating the node feature and path feature (Node+Path) achieves higher performance than the node feature and the path feature alone, as well as the baseline RDF2Vec which is based on graph walks. For DBP-Lit, it outperforms RDF2Vec by 39.9%, 44.1% and 45.5% for MRR, Hits@1 and Hits@5, respectively. Meanwhile, Node+Path performs better than TransE and DistMult for DBP-Lit, while for MED-Ent, TransE and DistMult outperforms Node+Path. For example, considering the metric of MRR, Node+Path is 71.3% higher than DistMult for DBP-Lit, but DistMul is 108.8% higher than Node+Path for MED-Ent. One potential reason is the difference in the number of properties and sparsity of the two sub-graphs. DBP-Lit has 127 properties in its target assertions and 1958 properties in its sub-graph; while MED-Ent has 7 properties in its target assertions and 19 properties in its sub-graph. The small number of properties for MED-Ent leads to quite poor path feature, which is verified by its independent performance (e.g., the MRR is only 0.09). In the sub-graph of DBP-Lit, the average number of connected entities per property (i.e., density) is 150.7, while in the sub-graph of MED-Ent, it is 2739.0. Moreover, a larger ratio of properties to entities also leads to richer path features. According to these results, we use Node+Path for DBP-Lit and DistMult for MED-Ent in our correction framework.

Regarding MUS-Map, the above observations of MED-Ent in comparing observed features and embedding models also hold: TransR and TransH outperform Path and Node+Path. As MED-Ent, MUS-Map also has a small number of object properties, and a higher average number of connected entities per property. A simple graph structure and a large number of connected entities per property make it easier to automatically learn the representations. What differs from MED-Ent is that Path alone is also quite effective, as the mapping properties of Music KG are symmetric and transitive, and they sometimes lead to a special path that can directly indicate the equality of two entities. For example, $\langle a_1, \textit{wd:musicbrainzArtist}, a_2\rangle$ is true if there is a path composed of two triples $\langle a_1, \textit{wd:discogsArtist}, a'\rangle$ and $\langle a_2, \textit{musicbrainz:discogsArtist}, a'\rangle$. A new observation from MUS-Map is that Node+Path+Str achieves the best performance. This is due to the additional predictive information from the entity names whose individual contribution has been verified by AttBiRNN. Note we do not measure Node+Path+Str for DBP-Lit and MED-Ent as the name similarity of the subject and

object is meaningless for judging the likelihood of a none-mapping assertion.

5.3.2. *Impact of The Sub-graph*

We further analyze the impact of using the sub-graph for training the assertion prediction model. Table 3 shows the results of some of the models that are trained with the whole KB, as well as the comparison against the results when they are trained with the sub-KB. On the one hand, in comparison with Node+Path trained purely with the sub-graph, Node+Path with features from the whole KB actually performs worse. As all the directly connected properties and entities of each subject entity, related entity and target property have already been included in the sub-graph, using the sub-graph makes no difference for node features and path features of depth one. Thus the above result is mainly due to the fact that path features of depth two actually makes limited contribution in this assertion prediction context. This is reasonable as they are weak, duplicated or even noisy in comparison with node features and path features of depth one. One the other hand, learning the semantic embeddings with the sub-graph has positive impact on TransE and negative impact on DistMult for MED-Ent. However the impact in both cases is quite limited. The results on MUS-Map also give similar observations which indicate that the sub-graph extracted can include the effective context for predicting the likelihood of candidate assertions.

The sub-graph keeps the accuracy, but significantly reduces the scale, as shown in Table 4. For example, the sub-graph of MED-Ent has only 36.9% (11.2% resp.) of the entities (assertions resp.) of the whole medical KB, and as a result it reduces the training time of DistMult embeddings from 46.7 minutes to 19.0 minutes. The reduction on the scale and embedding training for DBP-Lit and MUS-Map is even more significant. Note that the time of MLP training with the observed features is very little and mostly depends on the sample size, and thus does not differ from the whole KB to the sub-graph (we sample a fixed number of triples for training). The time of using observed features mostly lies in the extraction of node and path features, but it is can be very efficiently implemented by SPARQL queries even on the whole KB, thanks to efficient and scalable RDF reasoning engines such as RDFox [40] used in our evaluation.

5.4. *Overall Results*

Figure 4 presents the correction rate, empty rate and accuracy of our assertion correction framework with a ranging filtering threshold τ. Note that lexical matching without any filtering is close to the existing method discussed in related work [29]. On the one hand, we find that filtering with either assertion prediction (AP) or constraint-based validation (CV) can improve the correction rate. This is because those candidate substitutes that are lexically similar to the erroneous object but lead to unlikely assertions are filtered out, while those that are not so similar but lead to true assertions are ranked higher. As the empty rate is also definitely increased after filtering (e.g., improved from 0.252 to 0.867 by Lookup* + AP + CV for DBP-Lit), the accuracy of all three data sets is improved in the whole range of τ. On the other hand, we find that averaging the scores from assertion prediction and constraint-based validation is effective. It leads to both higher correction rate and accuracy than either of them for some values of τ, such as $[0.05, 0.1]$ for DBP-Lit and $[0.85, 0.95]$ for MED-Ent.

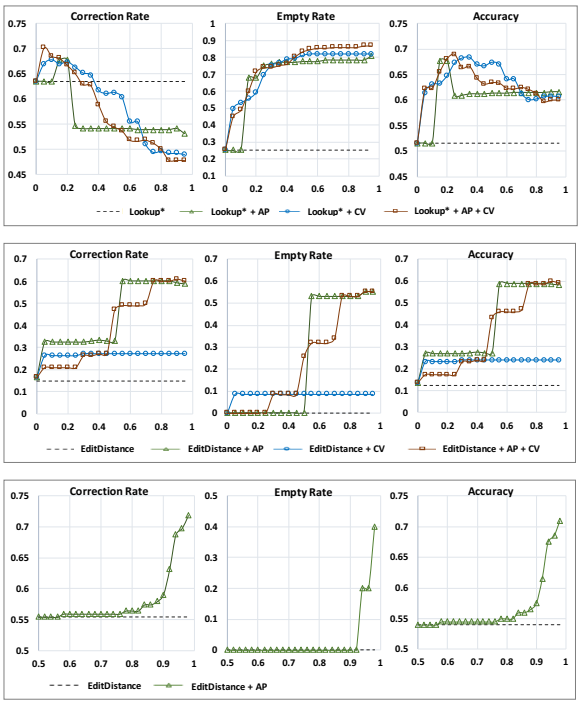

Fig. 4. Overall results of the correction framework for DBP-Lite [Above], MED-Ent[Middle] and MUS-Map [Below]. + AP and + CV represent filtering with assertion prediction and constraint-based validation respectively, with the filtering threshold $\boldsymbol{\tau}$ ranging from $\mathbf{0}$ to $\mathbf{1}$ with a step of $\mathbf{0.05}$.

Table 5 presents the optimum correction rate and accuracy for several settings. Note that they are achieved using a suitable τ setting; in real applications this can

Methods	DBP-Lit			MED-Ent			MUS-Map		
	MRR	Hits@1	Hits@5	MRR	Hits@1	Hits@5	MRR	Hits@1	Hits@5
Random	0.200	0.100	0.275	0.027	0.013	0.066	0.049	0.024	0.122
AttBiRNN	0.251	0.126	0.348	0.255	0.111	0.414	0.806	0.0721	0.903
RDF2Vec	0.419	0.320	0.492	—	—	—	—	—	—
TransE	0.392	0.266	0.507	0.744	0.652	0.862	0.596	0.546	0.604
TransH	0.508	0.377	0.639	0.748	0.656	**0.865**	0.824	0.799	0.851
TransR	0.510	0.391	0.632	0.738	0.640	0.823	0.842	0.805	0.883
DistMult	0.424	0.300	0.536	**0.752**	**0.694**	0.806	0.619	0.545	0.669
ComplEx	0.468	0.327	0.632	0.708	0.625	0.792	0.640	0.578	0.682
Node	0.495	0.379	0.604	0.338	0.171	0.514	0.004	0.006	0.006
Path	0.473	0.356	0.578	0.090	0.028	0.133	0.812	0.805	0.805
Node+Path	**0.586**	**0.461**	**0.716**	0.360	0.200	0.525	0.810	0.792	0.812
Node+Path+Name	—	—	—	—	—	—	**0.943**	**0.916**	**0.947**

Table 2

Results of the assertion prediction model trained by the sub-graph.

Cases	MRR	Hits@1	Hits@5	Training Time (min)
Node+Path (DBP-Lit)	0.504 (-0.082)	0.384 (-0.077)	0.611 (-0.105)	< 0.5 (≈ 0.0)
TransE (MED-Ent)	0.713 (-0.031)	0.608 (-0.044)	0.834 (-0.028)	28.0 (+19.1)
DistMult (MED-Ent)	0.766 (+0.014)	0.721 (+0.027)	0.822 (+0.016)	46.7 (+27.7)
TransH (MUS-Map)	0.769 (-0.055)	0.816 (+0.017)	0.848 (-0.003)	3846.3 (+3839.7)
Node+Path+Name (MUS-Map)	0.899 (-0.044)	0.902 (-0.014)	0.910 (-0.037)	< 0.5 (≈ 0.0)

Table 3

Results of the assertion prediction model trained by the whole KB, and the corresponding increase values w.r.t. the model trained by the sub-graph (the former minus the latter) in the parentheses.

Methods	DBP-Lit		MED-Ent		MUS-Map	
	Entities	Facts	Entities	Facts	Entities	Facts
Sub-graph	295,112	449,387	52,041	297,029	204,471	144,944
The Whole KB	$\approx$ 38.3 million	$\approx$ 480 million	233,035	939,400	$\approx$ 14.7 million	$\approx$ 90 million

Table 4

The scale of the sub-graph and the original whole KB.

be determined using a validation data set. With these results, we make the following observations. First, the optimum results are consistent with our above conclusions regarding the positive impact of assertion prediction, constraint-based validation and their ensemble. For example, the optimum accuracy of DBP-Lit is improved by 32.6% using constraint-based validation in comparison with the original related entities by lexical matching. The correction rate of MED-Ent provides another example: REE + AP + CV is 1.5% higher than REE + AP, and 121.4% higher than REE + CV.

Second, lexical matching using either Lookup (for DBP-Lit) or Edit Distance (for MED-Ent and MUS-Map) has a much higher correction rate and accuracy than that using *Word2Vec*, while our Lookup with sub-phrases (Lookup*) has even higher correction rate than the original Lookup of DBpedia. These overall results verify the recall analysis on related entity estimation in Section 5.2. Meanwhile, we find that the overall results on observed features ($\mathcal{M}_{np}$ and $\mathcal{M}_{nps}$) and latent semantic embeddings ($\mathcal{M}_{dm}$ and $\mathcal{M}_{tr}$) are also consistent with the assertion prediction analysis in Sec-

tion 5.3: $\mathcal{M}_{np}$ ($\mathcal{M}_{nps}$ resp.) has a better filtering effect than $\mathcal{M}_{dm}$ and $\mathcal{M}_{tr}$ for DBP-Lit (for MUS-Map resp.), but worse filtering effect for MED-Ent.

5.5. Constraint-based Validation

Besides the positive impact on the overall results mentioned above, we get several more detailed observations about constraint-based validation from Table 5. On the one hand, the property range constraint plays a more important role than the property cardinality constraint, while their combination is more robust than either of them, as expected. Considering the assertion set of MED-Ent, filtering by $\mathcal{M}_{ran}$, for example, leads to 62.6% higher accuracy than filtering by $\mathcal{M}_{car}$, while filtering by $\mathcal{M}_{ran+car}$ has 2.9% higher accuracy and equal correction rate in comparison with $\mathcal{M}_{ran}$.

On the other hand, we find constraint-based validation performs well for DBP-Lit, with higher accuracy and equal correction rate in comparison with assertion prediction, but performs much worse for MED-Ent. This is mainly due to the gap between the semantics of the two target assertion sets and their corresponding KBs: *(i)* the mined property ranges of DBP-Lit include 404 hierarchical classes, while those of MED-DB have only 8 classes in total and these classes have no hierarchy; *(ii)* 23 out of 127 target properties in DBP-Lit have pure functionality (i.e., $D^p_{car}(k = 1) = 1.0$) which plays a key role in the consistency checking algorithm, while none of the target properties of MED-Ent has such pure functionality. The second characteristic is also a potential reason why filtering by constraint-based validation with property cardinality only achieves a very limited improvement over Edit Distance for MED-Ent as shown in Table 5.

We additionally present some examples of the mined soft property constraints in Table 6. Most constraints such as the cardinality $1:1.000$ of the property *dbp:finalteam* are consistent with our common sense understanding of the properties, although some noise is evident, such as the range classes Person and Agent of the property *dbp:homeTown*). Most noise of the mined constraints is likely caused by erroneous property and class membership assertions in DBpedia.

6. Discussion and Outlook

In this paper, we present a study of assertion and alignment correction, an important problem for KB curation, but one that has rarely been studied. We have proposed a general correction framework, which does not rely on any KB meta data or external information, and which exploits both semantic embedding and consistency reasoning to correct erroneous objects, informally annotated literals (entity mentions) as well as wrong entity mappings in KB alignment. To improve the efficiency in dealing with very large KBs, it also includes techniques to extract candidate substitutes and sub-graphs. The framework and the adopted techniques have been evaluated by correcting assertions and mappings in three KBs: DBpedia with cross-domain knowledge, a medical KB and a music KB with artists from Wikidata, Musicbrainz and Discogs. We discuss below several more observations of the study and possible directions for the future work.

Entity relatedness. Our method follows the principle of correcting the object by a related entity rather than an arbitrary entity that leads to a correct assertion. This dramatically reduces the searching space of the correction and computation spent on assertion prediction. Relatedness can be due to either lexical or semantic similarity. Currently, the recall for DBP-Lit, MED-Ent and MUS-Map is 0.882, 0.797 and 0.785 respectively, which is promising, but still leaves space for further improvement. One extension for higher recall but limited noise and sub-graph size is incorporating more literal attributes and graph structure of the entity.

KB variation. Although both constraint-based validation and assertion prediction improve overall performance, their impact varies from DBpedia to the medical KB. The effectiveness of constraint-based validation depends on the richness of the KB schema, such as property functionality, the complexity of property range, etc. The more complex the schema is, the better performance constraint-based validation achieves. The impact of assertion prediction is more complicated: the path and node features perform better on DBpedia which has many more properties, while the semantic embeddings by DistMult and TransE are more suitable for the medical KB which has less properties but higher density. Integrating assertion prediction and constraint-based validation, even with simple score averaging, can improve performance, but further study is needed for a better integration method that is adapted to the given KB.

Property constraints. On the one hand, the evaluation indicates that the mined property constraints are effective for assertion validation and can be independently used in other contexts like online KB edit-

Methods	DBP-Lit		MED-Ent		MUS-Map	
	C-Rate	Acc	C-Rate	Acc	C-Rate	Acc
Lookup or Edit Distance	0.597	0.611	0.149	0.123	0.554	0.540
Lookup*	0.635	0.516	—	—	—	—
Word2Vec	0.553	0.410	0.089	0.076	0.426	0.415
REE + AP ($\mathcal{M}_{np}$ or $\mathcal{M}_{nps}$)	0.677	0.677	0.360	0.327	**0.718**	**0.710**
REE + AP ($\mathcal{M}_{tr}$ or $\mathcal{M}_{dm}$)	0.635	0.628	0.600	0.588	0.672	0.655
REE + CV ($\mathcal{M}_{ran}$)	0.671	0.668	0.271	0.239		
REE + CV ($\mathcal{M}_{car}$)	0.639	0.622	0.164	0.147	—	
REE + CV ($\mathcal{M}_{ran+car}$)	0.677	0.684	0.271	0.246		
REE + AP + CV	**0.701**	**0.690**	**0.609**	**0.599**	—	

Table 5

The correction rate (C-Rate) and accuracy (Acc). REE denotes Related Entity Estimation: DBP-Lit uses Lookup* while MED-Ent uses Edit Distance. MUS-Map does not adopt constraint-based validation.

Property	Cardinality	Specific Range	General Range
dbp: homeTown	1 : 0.415 2 : 0.422 3 : 0.163	Location: 0.801, City: 0.280, Country: 0.268, Person: 0.228, ...	PopulatedPlace: 0.821, Place: 0.821, Settlement: 0.299, Agent: 0.259, ...
dbp: finalteam	1 : 1.000	BaseballTeam: 0.493, SportsTeam: 0.154, SoccerClub: 0.015, ...	Agent: 0.688, Organization: 0.665, SportsTeam: 0.510, ...

Table 6

Soft constraints of two property examples of DBpedia

ing. On the other hand, unlike the assertion prediction model, the constraints as well as the consistency checking algorithm are interpretable. One benefit is that explicitly defined or external TBox constraints can easily be injected into our framework by overwriting or revising the constraints. For example, the mined specific range Person: 0.228 in Table 6, which is inappropriate for the property *dbp:homeTown*, can be directly removed. For another example, if the cardinality of *hasParent* is defined as 2, which is quite reasonable, its probability distribution can be directly set to $D^{p}_{car}(k) = 1.0$ if $k = 2$ and 0 otherwise.

Ontology alignment and repair. Traditional ontology alignment systems e.g., LogMap often rely on defined logic constraints to correct (usually eliminate) those mappings that lead to inconsistency. Recently they are also being extended to entity alignment for large KBs [20]. In the future work, we will evaluate our entity alignment correction solution on KBs with a richer schema (e.g., hierarchical classes and multiple properties) by analyzing the impact of constraint mining and comparing the results with the above ontology alignment systems.

Efficiency and scalability. The computation of the correction framework mainly lies in the following aspects: *(i)* related entity estimation which can be efficiently implemented by lexical indexes (e.g., the DBpedia Lookup service), *(ii)* SPARQL queries and KB reasoning for sub-graph extraction, node and path feature calculation, and property constraint mining, *(iii)* the training of assertion prediction models, especially the semantic embeddings, and *(iv)* the searching of the correction, i.e., the likelihood prediction and consistency validation of candidate assertions. SPARQL queries and reasoning can be supported by some efficient and scalable reasoning engines such as RDFox [40] used for our music KB. The computation for the last two aspects is dramatically reduced by using the related entities and the sub-graph, whose size depends on the related entity estimation algorithm used, and the data – the KB and the target assertions. We suggest to correct those related target assertions in one time for higher efficiency and a more complete context in the

sub-graph, although the method can work for an arbitrary set of target assertions.

Acknowledgments

This work was supported by the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project 237889), Samsung Research UK, Siemens AG, and the EPSRC projects AnaLOG (EP/P025943/1), OASIS (EP/S032347/1), UK FIRES (EP/S019111/1) and the AIDA project (Alan Turing Institute). Xi Chen's contribution and the GPU computation resources are supported by Jarvis Lab Tencent.

References

[1] Dörthe Arndt, Ben De Meester, Anastasia Dimou, Ruben Verborgh, and Erik Mannens. 2017. Using Rule-based Reasoning for RDF Validation. In *International Joint Conference on Rules and Reasoning*. Springer, 22–36.

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

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*. 2787–2795.

[4] Jiaoyan Chen, Xi Chen, Ian Horrocks, Ernesto Jimenez-Ruiz, and Erik B Myklebus. 2020. Correcting Knowledge Base Assertions. In *The Web Conference*. 1537–1547.

[5] Jiaoyan Chen, Ernesto Jiménez-Ruiz, and Ian Horrocks. 2019. Canonicalizing Knowledge Base Literals. In *International Semantic Web Conference*.

[6] Namyoun Choi, Il-Yeol Song, and Hyoil Han. 2006. A survey on ontology mapping. *ACM Sigmod Record* 35, 3 (2006), 34–41.

[7] Michalis Chortis and Giorgos Flouris. 2015. A Diagnosis and Repair Framework for DL-LiteA KBs. In *European Semantic Web Conference*. Springer, 199–214.

[8] Gerard De Melo. 2013. Not quite the same: Identity constraints for the web of linked data. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

[9] Jeremy Debattista, Christoph Lange, and Sören Auer. 2016. A Preliminary Investigation towards Improving Linked Data Quality Using Distance-based Outlier Detection. In *Joint International Semantic Technology Conference*. Springer, 116–124.

[10] Antonin Delpeuch. 2019. OpenTapioca: Lightweight Entity Linking for Wikidata. *arXiv preprint arXiv:1904.09131* (2019).

[11] Anastasia Dimou, Dimitris Kontokostas, Markus Freudenberg, Ruben Verborgh, Jens Lehmann, Erik Mannens, Sebastian Hellmann, and Rik Van de Walle. 2015. Assessing and Refining Mappings to RDF to Improve Dataset Quality. In *International Semantic Web Conference*. Springer, 133–149.

[12] John Domingue, Dieter Fensel, and James A Hendler. 2011. *Handbook of Semantic Web Technologies*. Springer Science & Business Media.

[13] Jérôme Euzenat. 2017. Interaction-based ontology alignment repair with expansion and relaxation. In *IJCAI 2017-26th International Joint Conference on Artificial Intelligence*. AAAI Press, 185–191.

[14] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. 2018. Linked Data Quality of DBpedia, Freebase, Opencyc, Wikidata, and Yago. *Semantic Web* 9, 1 (2018), 77–129.

[15] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. 2014. Canonicalizing Open Knowledge Bases. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, 1679–1688.

[16] Birte Glimm and Chimezie Ogbuji. 2013. SPARQL 1.1 Entailment Regimes. *W3C Recommendation* (2013).

[17] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter F. Patel-Schneider, and Ulrike Sattler. 2008. OWL 2: The Next Step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web* 6, 4 (2008), 309–322.

[18] Kalpa Gunaratna, Krishnaprasad Thirunarayan, Amit Sheth, and Gong Cheng. 2016. Gleaning Types for Literals in RDF Triples with Application to Entity Summarization. In *European Semantic Web Conference*. 85–100.

[19] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. OpenKE: An Open Toolkit for Knowledge Embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 139–144.

[20] Sven Hertling and Heiko Paulheim. 2020. The Knowledge Graph Track at OAEI–Gold Standards, Baselines, and the Golden Hammer Bias. *arXiv preprint arXiv:2002.10283* (2020).

[21] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. 2020. Knowledge Graphs. *arXiv preprint arXiv:2003.02320* (2020).

[22] Valentina Ivanova and Patrick Lambrix. 2013. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *Extended Semantic Web Conference*. Springer, 1–15.

[23] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. 2011. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*. Springer, 273–288.

[24] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks, and Rafael Berlanga. 2009. Ontology integration using mappings: Towards getting the right logical consequences. In *European Semantic Web Conference*. Springer, 173–187.

[25] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks, and Rafael Berlanga. 2011. Logic-based assessment of the compatibility of UMLS ontology sources. *Journal of biomedical semantics* 2, S1 (2011), S2.

[26] Evgeny Kharlamov, Bernardo Cuenca Grau, Ernesto Jiménez-Ruiz, Steffen Lamparter, Gulnar Mehdi, Martin Ringsquandl, Yavor Nenov, Stephan Grimm, Mikhail Roshchin, and Ian Horrocks. 2016. Capturing Industrial Information Models with Ontologies and Constraints. In *15th International Semantic Web Conference (ISWC)*. 325–343.

[27] Holger Knublauch and Dimitris Kontokostas. 2017. Shapes Constraint Language (SHACL). *W3C Recommendation* 20 (2017).

[28] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. 2014. Test-driven Evaluation of Linked Data Quality. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 747–758.

[29] Piyawat Lertvittayakumjorn, Natthawut Kertkeidkachorn, and Ryutaro Ichise. 2017. Correcting Range Violation Errors in DBpedia.. In *International Semantic Web Conference (Posters, Demos & Industry Tracks)*.

[30] Huanyu Li, Zlatan Dragisic, Daniel Faria, Valentina Ivanova, Ernesto Jiménez-Ruiz, Patrick Lambrix, and Catia Pesquita. 2019. User validation in ontology alignment: functional assessment and impact. *Knowl. Eng. Rev.* 34 (2019), e15.

[31] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[32] Christian Meilicke. 2011. *Alignment incoherence in ontology matching*. Ph.D. Dissertation. University of Mannheim. https://ub-madoc.bib.uni-mannheim.de/29351

[33] André Melo and Heiko Paulheim. 2017. An Approach to Correction of Erroneous Links in Knowledge Graphs. In *CEUR Workshop Proceedings*, Vol. 2065. RWTH, 54–57.

[34] André Melo and Heiko Paulheim. 2017. Detection of Relation Assertion Errors in Knowledge Graphs. In *Proceedings of the Knowledge Capture Conference*. ACM, 22.

[35] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems*. ACM, 1–8.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

[37] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhanava Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Commun. ACM* 61, 5 (2018), 103–115.

[38] Erik B. Myklebust, Ernesto Jiménez-Ruiz, Jiaoyan Chen, Raoul Wolf, and Knut Erik Tollefsen. 2019. Knowledge Graph Embedding for Ecotoxicological Effect Prediction. *CoRR* abs/1907.01328. arXiv:1907.01328 http://arxiv.org/abs/1907.01328

[39] Gonzalo Navarro. 2001. A Guided Tour to Approximate String Matching. *Comput. Surveys* 33, 1 (2001), 31–88.

[40] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. 2015. RDFox: A highly-scalable RDF store. In *International Semantic Web Conference*. Springer, 3–20.

[41] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, and Klaus Lyko. 2014. Unsupervised Link Discovery through Knowledge Base Repair. In *European Semantic Web Conference*. Springer, 380–394.

[42] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A Survey on Open Information Extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*. 3866–3878.

[43] Xing Niu, Xinruo Sun, Haofen Wang, Shu Rong, Guilin Qi, and Yong Yu. 2011. Zhishi.me - Weaving Chinese Linking Open Data. In *International Semantic Web Conference*. Springer, 205–220.

[44] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2018. Scalable Rule Learning via Learning Representation.. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2149–2155.

[45] Heiko Paulheim. 2017. Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. *Semantic web* 8, 3 (2017), 489–508.

[46] Heiko Paulheim and Christian Bizer. 2014. Improving the Quality of Linked Data Using Statistical Distributions. *International Journal on Semantic Web and Information Systems (IJSWIS)* 10, 2 (2014), 63–86.

[47] Heiko Paulheim and Aldo Gangemi. 2015. Serving DBpedia with DOLCE – More than Just Adding A Cherry on Top. In *International Semantic Web Conference*. Springer, 180–196.

[48] 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*. ACM, 1465–1475.

[49] Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1751–1756.

[50] Guilin Qi, Qiu Ji, and Peter Haase. 2009. A conflict-based operator for mapping revision. In *International Semantic Web Conference*. Springer, 521–536.

[51] Petar Ristoski and Heiko Paulheim. 2016. RDF2Vec: RDF Graph Embeddings for Data Mining. In *International Semantic Web Conference*. Springer, 498–514.

[52] Emanuel Santos, Daniel Faria, Catia Pesquita, and Francisco M Couto. 2015. Ontology alignment repair through modularization and confidence-based heuristics. *PloS one* 10, 12 (2015).

[53] Alessandro Solimando, Ernesto Jimenez-Ruiz, and Giovanna Guerrini. 2017. Minimizing conservativity violations in ontology alignments: Algorithms and evaluation. *Knowledge and Information Systems* 51, 3 (2017), 775–819.

[54] Alberto Tonon, Michele Catasta, Gianluca Demartini, and Philippe Cudré-Mauroux. 2015. Fixing the Domain and Range of Properties in Linked Data by Context Disambiguation. *LDOW@ WWW* 1409 (2015).

[55] Gerald Töpper, Magnus Knuth, and Harald Sack. 2012. DBpedia Ontology Enrichment for Inconsistency Detection. In *Proceedings of the 8th International Conference on Semantic Systems*. ACM, 33–40.

[56] Kristina Toutanova and Danqi Chen. 2015. Observed versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. 57–66.

[57] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. 2071–2080.

[58] Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. CESI: Canonicalizing Open Knowledge Bases Using Embeddings and Side Information. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1317–1327.

[59] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledge Base. *Commun. ACM* (2014).

[60] Peng Wang and Baowen Xu. 2012. Debugging ontology mappings: a static approach. *Computing and Informatics* 27, 1 (2012), 21–36.

[61] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[62] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

[63] Gabriel Weaver, Barbara Strickland, and Gregory Crane. 2006. Quantifying the Accuracy of Relational Statements in Wikipedia: A Methodology. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, Vol. 6. Citeseer, 358–358.

[64] Gerhard Weikum, Luna Dong, Simon Razniewski, and Fabian Suchanek. 2020. Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases. *arXiv preprint arXiv:2009.11564* (2020).

[65] Tien-Hsuan Wu, Zhiyong Wu, Ben Kao, and Pengcheng Yin. 2018. Towards Practical Open Knowledge Base Canonicalization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 883–892.

[66] Zhuang Yan, Li Guoliang, and Feng Jianhua. 2016. A survey on entity alignment of knowledge base. *Journal of Computer Research and Development* 1 (2016), 165–192.

[67] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv preprint arXiv:1412.6575* (2014).