City Research Online

City St George's, University of London

Citation: Chen, J., Chen, X., Horrocks, I., Jimenez-Ruiz, E. & Myklebus, E. B. (2020). Correcting Knowledge Base Assertions. In: WWW '20: Proceedings of The Web Conference 2020. (pp. 1537-1547). New York, NY, United States: Association for Computing Machinery. ISBN 978-1-4503-7023-3 doi: 10.1145/3366423.3380226

This is the published version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/23585/

Link to published version: https://doi.org/10.1145/3366423.3380226

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to City Research Online policy.

City Research Online: http://openaccess.city.ac.uk/ publications@citystgeorges.ac.uk

Correcting Knowledge Base Assertions

Jiaoyan Chen
University of Oxford
jiaoyan.chen@cs.ox.ac.uk

Xi Chen
Jarvis Lab Tencent
jasonxchen@tencent.com

Ian Horrocks
University of Oxford
ian.horrocks@cs.ox.ac.uk

Erik B. Myklebust
Norwegian Institute for Water Research
University of Oslo
erik.b.myklebust@niva.no

Ernesto Jimenez-Ruiz
City, University of London
University of Oslo
ernesto.jimenez-ruiz@city.ac.uk

ABSTRACT

The usefulness and usability of knowledge bases (KBs) is often limited by quality issues. One common issue is the presence of erroneous assertions, often caused by lexical or semantic confusion. We study the problem of correcting such assertions, and present a general correction framework which combines lexical matching, semantic embedding, soft constraint mining and semantic consistency checking. The framework is evaluated using DBpedia and an enterprise medical KB.

KEYWORDS

Knowledge Base Quality, Assertion Correction, Semantic Embedding, Constraint Mining, Consistency Checking

ACM Reference Format:
Jiaoyan Chen, Xi Chen, Ian Horrocks, Erik B. Myklebust, and Ernesto Jimenez-Ruiz. 2020. Correcting Knowledge Base Assertions. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380226

1 INTRODUCTION

Knowledge bases (KBs) such as Wikidata [42] and DBpedia [2] are playing an increasingly important role in applications such as search engines, question answering, common sense reasoning and data integration. However, they still suffer from various quality issues, including constraint violations and erroneous assertions [11, 31], that negatively impact their usefulness and usability. These 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% [44]), or may be introduced by the knowledge extraction process.

Existing work on KB quality issues covers not only error detection and assessment, but also quality improvement via completion, canonicalizaiton and so on [31]. Regarding error detection, erroneous assertions can be detected by various methods, including consistency checking with defined, mined or external constraints [18, 33, 38], prediction by machine learning or statistical methods [7, 22, 32], and evaluation by query templates [19]; see Section 2.1 for more details. However, the erroneous assertions are often eliminated [6, 27], and few robust methods have been developed to correct them.

Lertvittayakumjorn et al. [20] and Melo et al. [21] 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 KBs like the medical KB used in our evaluation. Both studies 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 non-existent, and again fail to capture contextual semantics.

Other work has focused on quality improvement, for example by *canonicalizing* assertions whose objects are literals that represent entities (i.e., entity mentions); 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 [29], 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 [15], 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 [15], which is useful for creating a new entity, and can sometimes identify candidate entities in the KB [4], but they do not propose a general correction method; see Section 2.2 for a more details.

In this paper, we propose a method for correcting assertions whose objects are either erroneous entities or literals. To this end, we have developed a general framework that exploits related entity estimation, link prediction and constraint-based validation, as shown in Figure 1. Given a set of target assertions (i.e., assertions that have been identified as erroneous), it uses semantic relatedness 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 a link prediction model

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '20, April 20–24, 2020, Taipei, Taiwan
© 2020 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380226

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 both erroneous entity assertions and literal assertions;
- It utilizes both semantic embeddings and observed features to capture the local context used for correction prediction, with a sub-graph extracted for higher efficiency;
- It complements the prediction with consistency against "soft" property constraints mined from the global KB;
- It evaluates the framework with erroneous entity assertions from a medical KB and literal assertions from DBpedia.

2 RELATED WORK

We survey related work of assertion validation which includes erroneous assertion detection, link prediction with semantic embeddings and observed features, canonicalization, and assertion correction.

2.1 Assertion Validation

The validity of KB assertions is clearly an important consideration. One way to is to check their 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. [38] enriched it with class disjointness, and property domain and range costraints, all derived via statistical analysis; Paulheim and Gangemi [33] enriched it via alignment with the DOLCE-Zero foundational ontology. Various constraint and rule languages such as Shapes Constraint Language (SHACL) [18], Rule-Based Web Logics [1] and SPARQL query templates [19], have also been proposed so that external knowledge can be encoded and applied.

As machine learning develops, various methods have been proposed to encode the semantics of entities and relations into vectors for prediction [43]. 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 [21]), autoencoder (e.g., RDF2Vec [36]), statistical distribution estimation (e.g., SDValidate [32]) 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. They can be directly used to estimate the assertion likelihood. State-of-the-art methods and implementations include DistMult [46], TransE [3], IterE [47], ComplEx [40], OpenKE [16] and so on. They can also be combined with algorithms such as outlier detection [7] and supervised classification [25] 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 [30] and suffer from low robustness when dealing with real world KBs that are often noisy and sparse [35].

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 [12, 41, 45]; CESI [41], 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 have no neighbourhood information (e.g., relationships with other entities) that can be utilized.

Chen et al. [4] and Gunaratna et al. [15] 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 [4] 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. [9] refined the mappings between Wikipedia data and DBpedia knowledge to correct errors during DBpedia construction, while Pellissier et al. [34] 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. Chortis et al. [5, 37] defined and added new properties to avoid violating integrity constraints, while Melo [6] removed *sameAs* links that lead to such violations. 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. [20] and Melo et al. [21] 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.

3 BACKGROUND

3.1 Knowledge Base

In this study we consider a KB that follows Semantic Web standards including RDF (Resource Description Framework), RDF Schema, OWL (Web Ontology Language)[1] and the SPARQL Query Language [10]. 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 domains and ranges. It may also use a more expressive language such as OWL to express constraints such as class disjointness, property cardinality and so on [14].

The ABox consists of a set of assertions (facts) describing concrete entities (individuals), each of which is represented by an Uniform Resource Identifier. 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, which is specified via triples using the *rdf:type* property. Sometimes we will use *class assertion* to refer to this latter kind of assertion and *property assertion* to refer to assertions where p is not a property from the reserved vocabulary or RDF, RDFS or OWL.

Such a KB can be accessed by SPARQL queries using a query engine that supports the relevant entailment regime (e.g., RDFS or OWL) [13]; 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 (aka knowledge graphs) 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 [23], while entities of Wikidata can be retrieved, for example, via the backend API of OpenTapioc [8].

3.2 Problem Statement

In this study, we focus on correcting ABox property assertions $\langle s, p, o \rangle$ where o is a literal (*literal assertion*) or an entity (*entity assertion*). Note that in the former case correction may require more than simple canonicalization; e.g., the property assertion $\langle$*Sergio_Agüero*, *playsFor*, *"Manchester United"*$\rangle$ should be corrected to $\langle$*Sergio_Agüero*, *playsFor*, *Manchester_City*$\rangle$.

Literal assertions can be identified by data type inference and regular expressions as in [15], while erroneous entity assertions can be detected either manually when the KB is applied in downstream applications or automatically by the methods discussed in Section 2.1. 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 often do not respect this constraint.

We assume that the input is a KB $\mathcal{K}$, and a set $\mathcal{E}$ of literal and/or entity assertions that have been identified as incorrect. For each 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 true, or reporting that there is no such an entity e in $\mathcal{K}$.

4 METHODOLOGY

4.1 Framework

As shown in Figure 1, our assertion correction framework mainly consists of related entity estimation, link prediction, constraint-based validation and correction decision making. Related entity estimation identifies those entities that are relevant to the object of the assertion. Given a target assertion $t = \langle s, p, o \rangle$, its related entities, ranked by the 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 link prediction and constraint-based validation over the candidate assertions.

Link prediction estimates the likelihood of each candidate assertion. For each entity e_i in RE_t, it considers a target assertion $t = \langle s, p, o \rangle$ and outputs a score that measures the likelihood of $\langle s, p, e_i \rangle$. To train such a link prediction model, a sub-graph that contains the context of the correction task (i.e., $\mathcal{E}$) 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 (TransE [3] and DistMult [46]), as well as some widely used observed features (path and node) are used to build the link 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 property cardinality or range constraints in the KB TBox,[3] but instead use mined constraints, each of which is associated with a supporting degree (probability).

[1] There is a revision of the Web Ontology Language called OWL 2, for simplicity we also refer to this revision as OWL.
[2] https://wiki.dbpedia.org/lookup

[3] 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.

Correction decision making combines the results of related entity estimation, link 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.

4.2 Related Entity Estimation

For each target assertion $t = \langle s, p, o\rangle$ in $\mathcal{E}$, related entity estimation directly adopts o as the input if o is a literal, or extracts the label of o if o is an entity. It returns a list RE_t containing up to k most related entities; i.e., $|RE_t| \leq k$. Our framework supports both a lexical matching based approach and a word embedding based approach; this allows us to compare the effectiveness of the two approaches on different KBs (see Section 5.2).

For those KBs with a lexical index, the lexical matching based approach can directly use a lookup service based on the index, which often returns a set of related entities for a given phrase. Direct lookup with the original phrase, however, often misses the correct entity, as the input phrase, either coming from the erroneous entity or the literal, is frequently noisy and ambiguous. For example, the DBpedia Lookup service returns no entities with the input "three gorges district" which refers to the entity *dbr:Three_Gorges_Reservoir_Region*. To improve recall, we retrieve a list of up to k entities by repeating entity lookup using sub-phrases, 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. The list of each lookup is ordered according to the relatedness (lexical similarity) to the original phrase, while all the lists are concatenated according to the above lookup order. 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. For those KBs without an existing lexical index, the lexical matching based approach adopts a string similarity score i.e., normalized Edit Distance [26] to calculate the relatedness of an entity with its label.

The word embedding based approach calculates the similarity of o against each entity in the KB, using vector representations of their labels (literals). It *(i)* tokenizes the phrase and removes the stop words, *(ii)* represents each token by a vector with a word embedding model (e.g., *Word2Vec* [24]) 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, and *(iv)* computes the distance-based similarity score of the two vectors by e.g., the cosine similarity.

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 might be quite different. Moreover, lexical matching with a lexical index makes it easy to utilize multiple items of textual information, such as labels in multiple languages and anchor text, where different names of an entity are often included.

4.3 Link Prediction

Given related entities RE_t of a target assertion $t = \langle s, p, o\rangle$, link 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. For efficiency in dealing with very large KBs, we first extract a multi-relational sub-graph for the context of the task, and then train the link prediction model with a sampling method, observed features and semantic embeddings.

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 part of $\mathcal{K}$, denoted as $\mathcal{K}_\mathcal{E} = \langle E, P, T\rangle$, where E denotes entities, P denotes object properties (relations) and T denotes assertions (triples). 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 assertions. Note that $\models$ means an assertion is either directly declared in the KB or can be inferred by the KB. The statements with $\models$ can be implemented by SPARQL: 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.

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⟩
```

4.3.2 Sampling. 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

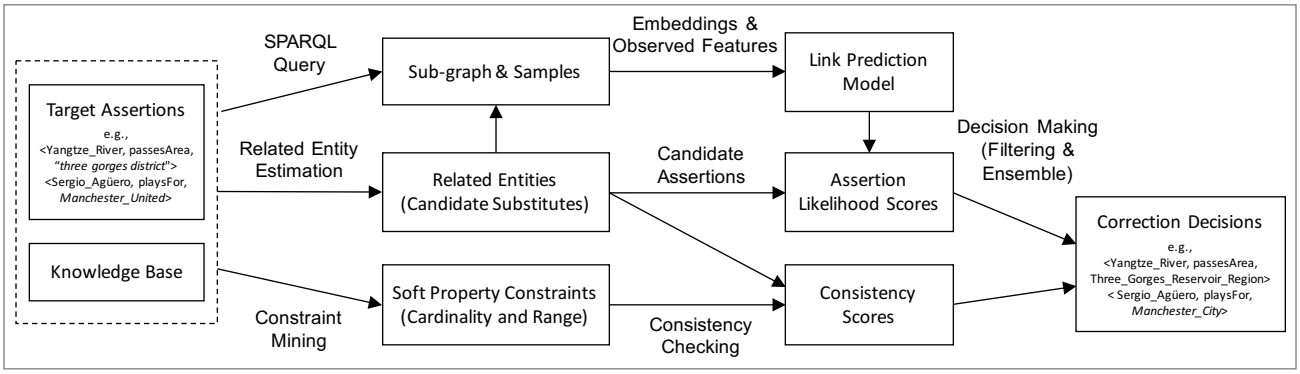

Figure 1: The Overall Framework for Assertion Correction

assertions whose objects and properties are among RE and P respectively ($T_{pr} = \{\langle s, p, o\rangle | p \in P \wedge o \in RE \wedge \langle s, p, o\rangle \in T\}$). 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 — 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 replacing 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 $\tilde{T}_{sp}$. 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 two kinds of observed features — the path feature and the node feature. The former represents potential relations that can connect the subject and object, while the latter represents the likelihood of the subject being the head of the property, and the likelihood of the object being the tail of the property. For the path feature, we limit the path depth to two, for reducing computation time and feature size, both of which are exponential 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 like DistMult and TransE, together with the node feature on some benchmarks [39]. The predictive information of a path also vanishes 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: $FP = FP^1_{so} \cup FP^1_{os} \cup FP^2_{so} \cup FP^2_{os}$, and encode them into a multi-hot vector as the path feature, denoted as f^p. Briefly we collect all the unique paths from the training assertions as a candidate set, where one path corresponds to one slot in encoding. When an assertion is encoded into a vector, a slot of the vector is set to 1 if the slot's corresponding path is among the assertion's paths 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.

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

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

We also adopt the path-based latent feature learned by the state-of-the-art algorithm RDF2Vec [36], 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, 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. We adopt two state-of-the-art algorithms — TransE [3] and DistMult [46]. Both are simple but have been shown to be competitive or outperform more complex alternatives [3, 17, 46]. For high efficiency, we learn the embeddings from the sub-graph.

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 that is configured, and are initialized by one-hot encoding. In learning, a batch stochastic gradient descent algorithm is used to minimize

the the following margin-based ranking loss:

$$L = \sum_{t \in T, t \rightarrow \tilde{t}} \left[\gamma + g(t) - g(\tilde{t})\right]_+ \quad (2)$$

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.

DistMult is a special form of the bilinear model where the non-diagonal entries in the relation matrices are assumed to be zero. The score function of an assertion $\langle s, p, o\rangle$ is defined as $\boldsymbol{e}_p^T(\boldsymbol{e}_s \circ \boldsymbol{e}_o)$, where $\circ$ denotes the operation of pairwise multiplication. As TransE, the embeddings are initialized by one-hot encoding, with the a dimension configured. A similar margin-based ranking loss as (2) is used for training with batched stochastic gradient descent.

In prediction, the likelihood score of an assertion can be calculated with the corresponding scoring function and the embeddings of its subject, property and object. We denote the link prediction model by TransE and DistMult as $\mathcal{M}_{te}$ and $\mathcal{M}_{dm}$ 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 $k \geq 1$ 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:

$$D^p_{car}(k) = \frac{|\{s \in S(p)|ON(s, p) = k\}|}{|S(p)|}, k = 1, ..., ON^p_{max}, \quad (3)$$

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 for $\sum_{i=n+1}^{ON^p_{max}} D^p_{car}(k = i)$, 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: $D_{car}(k = 1) = \frac{1}{10}$ and $D_{car}(k = 2) = \frac{9}{10}$. Note that although such constraints follow Closed Word Assumption and Unique Name Assumption, they behave well in 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 that the entity is an instance of according to class assertions in the KB. Note that there could be multiple such classes as the entity could be asserted to be an instance of multiple classes for which there is no sub-class relationship. General classes of an entity are those that subsume one or more of the specific classes as specified 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:

$$\begin{cases} D^p_{sp}(c) = \frac{|\{oe|oe \in OE(p), c \in C_{sp}(p, oe)\}|}{|OE(p)|}, c \in C^p_{sp}, \\ D^p_{ge}(c) = \frac{|\{oe|oe \in OE(p), c \in C_{ge}(p, oe)\}|}{|OE(p)|}, c \in C^p_{ge}. \end{cases} \quad (4)$$

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, 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 $1 \leq n \leq ON^p_{max} + 1$. 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 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 $r \geq \sigma$. Specially, when σ is set to 1.0, $r \geq \sigma$ (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 combing 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 letter 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 a link prediction model ($\mathcal{M}_{pn}$, $\mathcal{M}_{te}$ or $\mathcal{M}_{dm}$), 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. It eventually returns none, which means there is no entity in the KB that can replace the object of t, if RE'_t is empty, and the top-1 entity in RE'_t as the object substitute otherwise. The ensemble of the link 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 (cf. Section 5.4).

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

1 **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: $\langle \omega_c, \omega_g \rangle$

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

3 **begin**

4 *% count the number of object entities*

5 $n = |\{o | \mathcal{K} \models \langle s, p, o \rangle, o \text{ is entity}\} \cup \{e\}|$;

6 $r = \frac{(n - ON^p_{max})}{ON^p_{max}}$; *% calculate the exceeding rate*

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

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

9 $y_{car} = 0$;

10 **else**

11 **if** $n = 1$ **then**

12 *% probability as a functional property*

13 $y_{car} = D^p_{car}(k = 1)$;

14 **else**

15 *% probability as a none-functional property*

16 $y_{car} = \begin{cases} D^p_{car}(k > 1), & \text{if } r \leq 0 \\ (1 - r) \cdot D^p_{car}(k > 1), & \text{else} \end{cases}$

17 $C(e) = \{c | \mathcal{K} \models \langle e, \textit{rdf:type}, c \rangle\}$; *% get the object's classes*

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

19 $\begin{cases} y_{ran,c} = 1 - \prod_{c \in C^p_{sp} \cap C(e)} (1 - D^p_{ran}(c)), \\ y_{ran,g} = 1 - \prod_{c \in C^p_{ge} \cap C(e)} (1 - D^p_{ran}(c)); \end{cases}$

20 *% calculate the overall range constraint score*

21 $y_{ran} = \omega_c \cdot y_{ran,c} + \omega_g \cdot y_{ran,g}$;

22 **return** y_{car}, y_{ran}

5 EVALUATION

5.1 Experiment Settings

5.1.1 Data. In our experiment, we correct assertions in DBpedia [2] and 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 [28]). DBpedia is accessed via its official Lookup service, SPARQL Endpoint[4] 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 data are representative of two common situations: errors of DBpedia are mostly inherited from the source while errors of the medical KB are mostly introduced by extraction.

For DBpedia, we reuse the literal set proposed by [4, 15]. 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, we use a set of assertions with erroneous entity objects that have been discovered and collected during deployment of the KB in enterprise products; the GT annotations have been added with the help of clinical experts. For convenience, we call the above two target assertion sets DBP-Lit and MED-Ent respectively.[5] More details are shown in Table 1.

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

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

	Assertions (with Entity GT) #	Properties #	Subjects #
DBP-Lit	725 (499)	127	668
MED-Ent	272 (225)	7	200

Table 1: Some statistics of DBP-Lit and MED-Ent.

5.1.2 Settings. In the evaluation, we first analyze related entity estimation (Section 5.2) and link prediction (Section 5.3) independently. For the former, we report the recall of Entity GTs of different methods with varying top-k values, based on which suitable settings are selected for the framework. For link prediction, we compare different KB embeddings and observed features, 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) are calculated.

Next we evaluate the overall results of the assertion correction framework (Section 5.4), where the baselines are compared with, and the impact of link 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 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.

The reported results are based on the following setting (unless otherwise specified). *Word2Vec* [24] is trained using the Wikipedia article dump in June 2018. In link prediction, γ is set by linear increasing w.r.t. the training step, the embedding size is set to 100, and the other training hyper parameters such as the number of epochs and MLP hidden layer size are set such that the highest MRR is achieved on an evaluation sample set. For RDF2Vec, pre-trained versions of DBpedia entities by Mannheim University[6] 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 achieve similar results. The embeddings are trained by GeForce GTX 1080 Ti with OpenKE [16], while the remaining is computed by Intel(R) Xeon(R) CPU E5-2670 @2.60GHz.

5.2 Related Entity Estimation

We compare different methods and settings used in related entity estimation with the results presented in Figure 2, where the recall of Entity GTs by top-k related entities are presented. First, we find that the lexical matching based methods (Lookup, Lookup* and Edit Distance) have much higher recall than *Word2Vec*, on both 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 anchor text 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. 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. According to the absolute value and the trend of the recall curves in Figure 2, our framework uses Lookup* with $k = 30$ for DBP-Lit, and Edit Distance with $k = 76$ for MED-Ent.

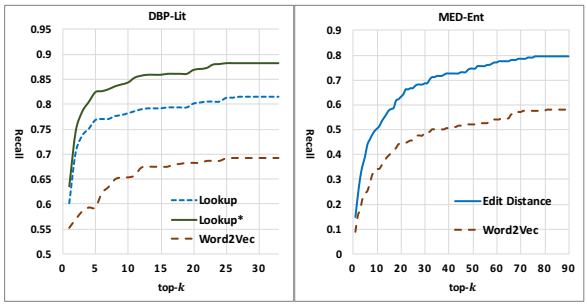

Figure 2: The recall of Entity GTs by top-k related entities

5.3 Link Prediction

5.3.1 Impact of Models. The results of different link 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, which is successfully applied in [4] for literal object typing, refers to the attentive bidirectional Recurrent Neural Networks that utilize the labels of the subject, property and object. First of all, the results verify that either latent semantic embeddings or observed features with MLP are effective for both DBP-Lit and MED-Ent: MRR, Hits@1 and Hits@5 are all dramatically improved w.r.t. Random and AttBiRNN.

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

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

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.

Methods	DBP-Lit			MED-Ent		
	MRR	Hits@1	Hits@5	MRR	Hits@1	Hits@5
Random	0.199	0.100	0.275	0.027	0.013	0.066
AttBiRNN	0.251	0.126	0.348	0.255	0.111	0.414
TransE	0.342	0.173	0.518	0.744	0.652	**0.862**
DistMult	0.424	0.300	0.536	**0.752**	**0.694**	0.806
RDF2Vec	0.419	0.320	0.492	—	—	—
Node	0.495	0.379	0.604	0.338	0.171	0.514
Path	0.473	0.356	0.578	0.090	0.028	0.133
Node+Path	**0.586**	**0.461**	**0.716**	0.360	0.200	0.525

Table 2: Link prediction results based on the sub-graph.

5.3.2 Impact of The Sub-graph. We further analyze the impact of sub-graph usage in training the link prediction model. The results of some of the methods that can be run over the whole KB with limited time are shown in Table 3, where Node+Path (DBP-Lit) uses features extracted from the whole KB but samples from the sub-graph. One the one hand, in comparison with Node+Path trained purely with the sub-graph, Node+Path with global features actually performs worse. As all the directly connected properties and entities of each subject entity, related entity and target property are 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 link 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. Considering that the sub-graph has only 36.9% (11.2% resp.) of the entities (assertions resp.) of the whole medical KB, which reduces the training time of DistMult embeddings from 46.7 minutes to 19.0 minutes, the above performance drop can be accepted.

Cases	MRR	Hits@1	Hits@5
TransE (MED-Ent)	0.713 (-.031)	0.608 (-.044)	0.834 (-.028)
DistMult (MED-Ent)	0.766 (+.014)	0.721 (+.027)	0.822 (+.016)
Node+Path (DBP-Lit)	0.504 (-.082)	0.384 (-.077)	0.611 (-.105)

Table 3: Link prediction results based on the whole KB, and their outperformance (gap) over those based on the sub-graph

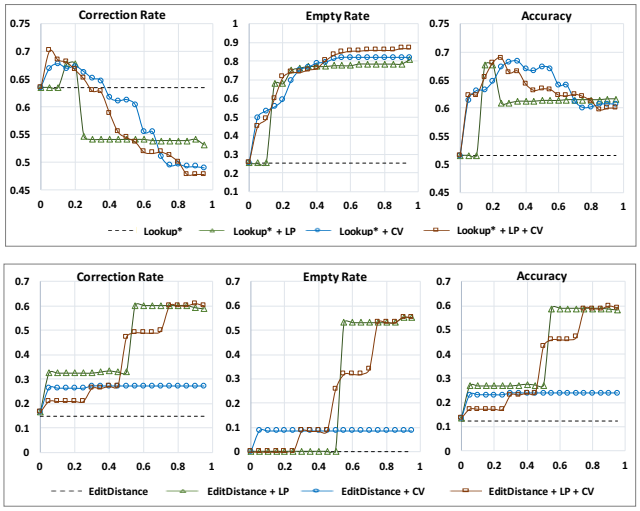

Figure 3: Overall results of the correction framework for DBP-Lite [Above] and MED-Ent [Below]. + LP and + CV represent filtering with link prediction and constraint-based validation respectively, with the filtering threshold τ ranging from 0 to 1 with a step of 0.05.

5.4 Overall Results

Figure 3 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 similar to the existing method discussed in related work [20]. On the one hand, we find that filtering with either link prediction (LP) or constraint-based validation (CV) can improve the correction rate when τ is set to a suitable range. 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 lexically similar but lead to true assertions are ranked higher. As the empty rate is definitely increased after filtering (e.g., improved from 0.252 to 0.867 by Lookup* + LP + CV for DBP-Lit), the accuracy for both DBP-Lit and MED-Ent is improved in the whole range of τ. On the other hand, we find that averaging the scores from link 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.

Table 4 presents the optimum correction rate and accuracy. Note that they are achieved using a suitable τ setting; in real applications this can be determined using an evaluation 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 link 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 + LP + CV is 1.5% higher than REE + LP, and 121.4% higher than REE + CV.

Second, lexical matching using either Lookup (for DBP-Lit) or Edit Distance (for MED-Ent) has a much higher correction rate and accuracy than *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 latent semantic embeddings ($\mathcal{M}_{dm}$ and $\mathcal{M}_{te}$) are also consistent with the link prediction analysis in Section 5.3: $\mathcal{M}_{np}$ has a better filtering effect than $\mathcal{M}_{dm}$ and $\mathcal{M}_{te}$ for DBP-Lit, but worse filtering effect for MED-Ent.

Methods	DBP-Lit		MED-Ent	
	C-Rate	Acc	C-Rate	Acc
Lexical Matching	0.597	0.611	0.149	0.123
Lookup*	0.635	0.516	—	—
Word2Vec	0.553	0.410	0.089	0.076
REE + LP ($\mathcal{M}_{np}$)	0.677	0.677	0.360	0.327
REE + LP ($\mathcal{M}_{dm}$)	0.635	0.628	0.600	0.588
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 + LP + CV	**0.701**	**0.690**	**0.609**	**0.599**

Table 4: Optimum correction rate (C-Rate) and accuracy (Acc). REE denotes Related Entity Estimation: Lookup* for DBP-Lit, Edit Distance 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 4. 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 link prediction, but performs 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 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 4.

We additionally present some examples of the mined soft property constraints in Table 5. Most of them are consistent with our common sense understanding of the properties, although some noise is evident (e.g., the range Person and Agent of *dbp:homeTown*), most likely caused by erroneous property and class assertions.

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 5: Soft constraints of two property examples

6 DISCUSSION AND OUTLOOK

In this paper we present a study of assertion correction, an important problem for KB curation, but one that has rarely been studied. We have proposed a general correction framework, which rely on no KB meta data or external information, and which exploits both deep learning and consistency reasoning to correct erroneous objects and informally annotated literals. The framework has been evaluated by correcting assertions in two KBs: DBpedia with cross-domain knowledge and an medical KB. We discuss below several more observations 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. Relatedness can be due to either lexical or semantic similarity. Currently, the recall for DBP-Lit and MED-Ent is 0.882 and 0.797 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 of the entity.

KB variation. Although both constraint-based validation and link 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 link 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 link 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 editing. On the other hand, unlike the link 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 5, which is inappropriate for the property *dbp:homeTown*, can be directly removed.

ACKNOWLEDGMENTS

This work was supported by the AIDA project (Alan Turing Institute), the SIRIUS Centre for Scalable Data Access (Research Council of Norway), Samsung Research UK, Siemens AG, and the EPSRC projects AnaLOG (EP/P025943/1), OASIS (EP/S032347/1) and UK FIRES (EP/S019111/1). Data of MED-Ent, GPU resources and Xi Chen's contribution were 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, Ernesto Jiménez-Ruiz, and Ian Horrocks. 2019. Canonicalizing Knowledge Base Literals. In *International Semantic Web Conference*.

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

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

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge Base Completion: Baselines Strike Back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. 69–74.

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

[19] 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.

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

[21] 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.

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

[23] 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.

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

[25] 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

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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).

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledge Base. (2014).

[43] 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.

[44] 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.

[45] 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.

[46] 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).

[47] Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. In *The World Wide Web Conference*. ACM, 2366–2377.