An Approach to Correction of Erroneous Links in Knowledge Graphs

André Melo
University of Mannheim
B6 26, 68161 Mannheim, Germany
andre@informatik.uni-mannheim.de

Heiko Paulheim
University of Mannheim
B6 26, 68161 Mannheim, Germany
heiko@informatik.uni-mannheim.de

ABSTRACT

Enriching and cleaning datasets are tasks relevant to most large-scale knowledge graphs, which are known to be noisy and incomplete. In this paper, we propose one approach to fix wrong facts which originate from confusion between entities. This is a common source of errors in Wikipedia, from which DBpedia and YAGO are extracted, and in open information extraction systems, e.g. NELL. Our approach generates candidate subjects and objects to fix triples by exploiting disambiguation links and approximate string matching. We run our approach on DBpedia and NELL and perform a manual evaluation of the generated corrections.

CCS CONCEPTS

• **Information systems → Data cleaning**; • **Computing methodologies → Semantic networks**; **Statistical relational learning**; **Machine learning**;

KEYWORDS

Knowledge graphs completion, error detection, data quality

ACM Reference format:
André Melo and Heiko Paulheim. 2017. An Approach to Correction of Erroneous Links in Knowledge Graphs. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Knowledge graphs are known to be both often incomplete and incorrect. Several link prediction and error detection methods have been proposed, however, few of them explicitly focus on error correction or address the problem of choosing which absent facts should be added to the knowledge graph.

The problem is that the number of possible relation assertions grows quadratically with the number of instances $n_c = n_i^2 n_r - n_f$, where n_i is the number of instances, n_r the number of relations and n_f the number of existing facts in the graph. For large datasets such as DBpedia, Wikidata and YAGO, computing the confidence score of all these facts is challenging. While pruning possible facts which violate ontology constraints, especially domain and range restrictions of relations, can significantly reduce the search space, the problem is still very challenging. To illustrate the size of the search space, in DBpedia (2016-10) $n_c \approx 4.4 \times 10^{17}$ facts; when filtering those triples which violate the domain and range restriction the number is reduced to $n_c \approx 2.8 \times 10^{17}$, which is still too large to compute confidence for all those candidates.

A promising approach for enriching a KG with some of its missing facts is the correction of erroneous facts. Some of the wrong facts which exist in a KG can be corrected. The error originates from some problem in the knowledge acquisition process, or in the source data. A good example of the latter are errors in Wikipedia, which serves as the main source of DBpedia and YAGO. If any of the links in the infobox are wrong, the extraction process generates a wrong fact. NELL, on the other hand, has text as main source of information and in many cases the source text has correct information, which cannot be extracted correctly.

In both cases it is common that an instance is confused with another one of a similar name (i.e., label or IRI). For example, the fact `formerTeam( Alan_Ricard, Buffalo_Bill)` is an erroneous fact from DBpedia which originates from a typo in Wikipedia: when referring to the NFL team `Buffalo_Bills`, the *s* was missing, therefore, the NFL team was confused with the character `Buffalo_Bill`. In NELL the entity `insect_raccoon` exists because of problems when extracting the fact that raccoons prey on insects, and is confused with `mammal_raccoon`.

Some relation assertion error detection approaches, such as PaTyBRED [4] and SDValidate [13], rely on type information, and since erroneous type assertions are also a common problem, that might result in correct relation assertions with an instance of incorrect or incomplete types being wrongly identified as erroneous. Therefore, combining such methods with type prediction [6, 12] is beneficial to rule out cases where the error is detected rather due to a missing or incorrect type of the subject or object than due to an erroneous relation assertion.

Therefore, it is relevant to make a careful analysis of detected errors, identify the source of each error, and if possible correct them. In this paper, we propose CoCKG (Correction of Confusions in Knowledge Graphs), an automatic correction approach which resolves relation assertion errors caused by instance confusion. The approach relies on error detection methods as well as type predictors to asses the confidence of the corrected facts. It uses approximate string matching and exploits both searching for entities with similar IRIs as well as Wikipedia disambiguation pages (if available) to find candidate instances for correcting the facts.

2 RELATED WORK

In the knowledge graph context, there are mainly error detection and link prediction approaches. Both are closely related to our

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

problem: while error detection deletes wrong triples, link prediction aims at adding new triples. In both cases, the approaches learn a KG model which is capable of assigning confidence values to triples.

KG embedding models [1, 10, 14] are currently the best performing approaches in the link prediction task. Other models rely on directly observed features, e.g., the path ranking algorithm (PRA) [3]. A more detailed description of link prediction methods can be found in [9]. It is important to note that none of the link prediction approaches mentioned address the problem of covering the candidate triples space (of size n_c as discussed in the introduction). Our approach, on the other hand, exploits the assumption that erroneous facts often have a corresponding correct fact in order to reduce that space. Error detection approaches, such as SDValidate and PaTyBRED, focus on the detecting of already existing erroneous triples. It has been shown that state-of-the-art embeddings perform worse than PaTyBRED in the error detection task [5]. A survey covering various KG refinement methods can be found in [11].

Rule-based systems, such as AMIE [2], cannot assign scores to arbitrary triples. However, they could be used to restrict the n_c search space by identifying high confidence soft rules and using the missing facts from instances where the rule does not hold as candidates. Combining them with previously mentioned KG models would be an interesting line of reasearch, however, it is out of the scope of this paper.

Wang et al. [15] studied the problem of erroneous links in Wikipedia, which is also the source of many errors of DBpedia. They model the Wikipedia links as a weighted directed mono-relational graph, and propose the LinkRank algorithm which similar to PageRank, but instead of ranking the nodes (entities), it ranks the links. They use LinkRank to generate candidates for the link correction and use textual features from the description of articles to learn a SVM classifier that can detect errors and choose the best candidate for correction. While this is a closely related problem, which can help mitigate the problem studied in this paper, their method cannot be directly applied on arbitrary knowledge graphs. Our approach takes advantage of the multi-relational nature of KGs, entity types, ontological information and the graph structure.

3 PROPOSED APPROACH

Our approach consists of first running an error detection algorithm (PaTyBRED in the case of this paper), selecting the top-k facts most likely to be wrong. In the next step, the error is heuristically verified to be an actual relation assertion error and not caused by missing type assertions in the object or subject with a type predictor tp. In the final step, candidate entities are retrieved, and if any of the candidates significantly improves the likelihood of the triple being right, we replace it by that candidate. The function CORRECT_TRIPLE in Algorithm 1 gives an overview of how CoCKG works. The parameter $\mathcal{K}$ is the set of all triples in the knowledge graph, $\mathcal{T}_{err}$ is the set of triple and confidence pairs generated by the error detection model (ed), tp is the type predictor, mc is the minimum confidence threshold, and mcg the minimum confidence gain threshold, i.e. the ratio of the new and old triple scores. In the next subsections we discuss the other parts in more details.

Algorithm 1 Knowledge base correction process

```
1: function CORRECT_TRIPLES(K, T_err, ed, tp, mc, mcg)
2:     T_corr ← ∅
3:     for t, score_t ∈ T_err do
4:         s, p, o ← t
5:         s_tp ← PREDICT_TYPES(tp, s)
6:         o_tp ← PREDICT_TYPES(tp, o)
7:         if ¬(CONF_NT(ed, t, s, s_tp) ∨ CONF_NT(ed, t, o, o_tp)) then
8:             s_cand ← GET_CANDIDATES(s)
9:             o_cand ← GET_CANDIDATES(o)
10:            T_cand ← {(s_i, p, o)|s_i ∈ s_cand} ∪ {(s, p, o_i)|o_i ∈ o_cand}
11:            T_cand ← T_cand − K
12:            c_best, max_conf ← nil, conf
13:            for c ∈ T_cand do
14:                if s ∈ domain(p) ∧ o ∈ range(p) then
15:                    score_c ← CONF(ed, c)
16:                    if score_c ≥ mc ∧ score_c/score_t ≥ mcg then
17:                        c_best, max_conf ← c, score_c
18:                    end if
19:                end if
20:            end for
21:            if c_best ≠ nil then
22:                T_corr ← T_corr ∪ {(c_best, t)}
23:            end if
24:        end if
25:    end for
26:    return T_corr
27: end function
```

3.1 Type Prediction

After selecting the k triples most likely to be wrong, we first check if their confidence is low because of missing or wrong instance types (subject or object). In order to do that, we run a type predictor tp on the subject and object instances. In this paper, we use as tp a multilabel random forest classifier based on qualified links (i.e. ingoing links paired with subject type and outgoing links paired with object type), as described in [6]. If the set of predicted types of the subject are different from the actual types, we change the type features used by ed and compute a new confidence for the triple (c.f. CONF_NT). If the new score satisfies mc and mcg, then we conclude that the error was in the subject type assertions. The same is done for the object, and if in neither case the confidence thresholds are satisfied, we proceed to the next part where we try to substitute the subject and object with their respective lists of candidates.

Combining the type prediction process with the error detection also has the advantage that the newly predicted types can be validated on triples containing the instance whose types were predicted. This can help support, or contradict the type predictor, possibly detecting types which are wrongly predicted by identifying triples where the score is lowered with the new types.

3.2 Retrieving Candidates

One simple way to find candidate entities to resolve entity confusions is to use the disambiguation links. Since disambiguation pages are only available for Wikipedia-based knowledge graphs, and furthermore are not available for each entity (e.g. `Ronaldo` has no disambiguation page), and in some cases the disambiguation pages miss important entities (e.g. the page `Bluebird_(disambiguation)` misses the entity `Bluebird_(horse)`, hence, we cannot correct the fact `grandisre(Miss_Potential,Bluebird)`), we require an additional source of candidates.

Since in our experiments we consider DBpedia and NELL, which have informative IRIs (in the case of DBpedia extracted from the correspondent Wikipedia's page), we search for candidate entities which have similar IRIs. Alternatively, it could also be done with entity labels. This would be useful in KGs which have uninformative IRIs (e.g. Wikidata and Freebase). For simplicity, in this paper, we refer to the informative part of an IRI as the "name" of the entity.

Retrieving all the instances of similar names can be a complicated task. This kind of problem is known as approximate string matching, and it has been widely researched [8, 16]. For our method we use an approximate string matching approach based on [7]. First, we remove the IRI's prefix and work with the suffix as the entity's name. We then tokenize the names and construct a deletions dictionary with all tokens being added with all possible deletions up to a maximum edit distance d_{max} threshold. This dictionary contains strings as keys and lists with all tokens which can turn into the key string with up to d_{max} deletions as values. Only pairs of tokens which share a common deletion string can have an edit distance less or equal than d_{max}. We also have a tokens dictionary which has tokens as keys and lists of entities which contain a given token as values. With that, given a token and a d_{max} we can easily obtain all the entities which contain that a string approximately similar to that token up to the maximum edit distance.

When searching for entities similar to a given entity, we perform queries for every token of the entity's name and we require that all tokens are matched. That is, for a certain entity to be considered similar, it has to contain tokens similar to all the tokens of the queried entity. A retrieved entity may have more tokens than the queried entity, but not less. The idea is that in general, when entering an entities name manually (e.g., in Wikipedia), it is common to underspecify the entity, but highly unlikely to overspecify it. E.g., it is more likely that `Ronaldo` is wrongly used instead of `Cristiano_Ronaldo` than the other way around. Furthermore, it reduces the number of matched entities.

We also perform especial treatment on DBpedia and NELL entity names because of peculiarities in their IRI structures. In DBpedia it is common to have between parentheses information to help disambiguate entities, which we consider unnecessary since the entity types are used in the error detection method. In NELL the first token is always the type of the entity, therefore, for similar reasons, we ignore it.

3.3 Correcting Wrong facts

At this point, for each assertion identified as erroneous, we have our list of candidate instances for subject and object from the disambiguation links and approximate string matching. We then compute a custom similarity measure $s(e_1, e_2)$ between an entity e_1 and a candidate e_2. Each entity e_i consists of a set of its tokens. The measure we propose consists of two components. The first is the sum of Levenshtein (d_L) distance of all matched tokens, and the second considers the number of unmatched tokens to capture a difference in specificity. The set of approximately matched token pairs is represented by $\mu(e_1, e_2)$ and the constant c is the weight of the second component. This measure is used to sort the retrieved candidates, to prune them in case there are too many, and to break ties when deciding which of the top-scoring candidates should be chosen.

$$s(e_1, e_2) = \sum_{(t_1, t_2) \in \mu(e_1, e_2)} d_L(t_1, t_2) + c \frac{|e_1| - |\mu(e_1, e_2)|}{|e1|} \quad (1)$$

In case the relation has domain or range restrictions, we remove the candidates which violate these restrictions. Later, for each of the candidates, we generate triples by substituting the subject and object by each of the instances in its candidates lists (first substitute subject only, then object only). That is, the total number of candidate triples is the sum of the size of the subject and object candidates list. We do not create candidate triples by substituting both the subject and object at the same time because, although possible, we assume the simultaneous confusion of both instances to be highly unlikely.[1] This is also done in order to make the number of candidate triples linear instead of quadratic.

We then remove the candidate triples which are already existent in the KG. We compute the confidence of all candidate triples and select that with highest confidence, given that *mc* and *mcg* are satisfied. Our method then outputs a list of triple pairs containing the wrong triple detected and the corrected triple predicted.

4 EXPERIMENTS

In our experiments we run CoCKG on DBpedia (2016-10) and NELL (08m-690), then we manually evaluate the triples corrected by our approach. We run PaTyBRED on both datasets and select top-1% facts most likely to be errors to be processed by our correction method. We classify each corrected fact in four different categories:

(1) WC: wrong fact turned into correct
(2) WW: wrong fact turned into another wrong fact
(3) CW: correct fact turned into wrong fact
(4) CC: correct fact turned into another correct fact

Our approach was run with $mc = 0.75, mcg = 2$ and entity similarity measure with $c = 1.5$[2]. That resulted in 24,973 corrections on DBpedia and 616 correction on NELL. It also detected that 873 (569) errors were caused by wrong types in DBpedia (NELL). Since manually evaluating all these corrections would be impossible, we randomly select 100 correction on each to perform the evaluation.

The results of our manual evaluation are shown in Figure 1. The proportion of facts successfully corrected (case 1) was rather low. While our approach can potentially improve the results by tweaking the parameters, and possibly using ensembles of different type predictors and error detectors, it currently cannot be used as a fully automatic approach. However, we believe that combining our approach with active learning is a promising direction which, with the help of specialists, could significantly improve results.

When evaluating some relations individually, we notice that some of them achieve good results. E.g., the relations `sire`, `damsire`, `grandsire` and `subsequentWork` reaching more than 90% of successful corrections (case 1). The results are good for these relations because horses are often named after other entities and artists often have albums named after themselves, which makes confusions easy to happen.

[1] For that to happen in the case of DBpedia, a Wikipedia user would have to go to the wrong article page and insert a wrong link in the infobox.
[2] The parameter values were selected based on heuristics and may not be optimal

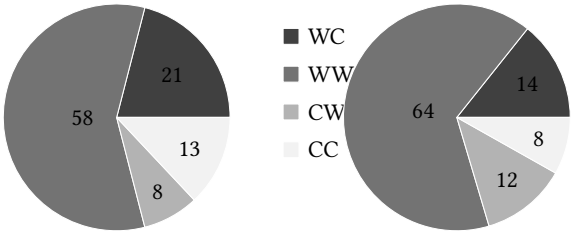

Figure 1: Manual evaluation on DBpedia and NELL respectively

One of the problems of our approach is that since it relies on PaTyBRED, which cannot find many relevant path features on DBpedia and NELL [5], it is difficult to distinguish between candidate entities of same type. For example, in NELL, the entity `person_paul` as object of `book_writer` relation is always corrected with `writer_paul_feval`.

The decision to generate candidate triples by corrupting either the subject or object seemed to have worked well for DBpedia, where we could not find a triple where both subject and object were wrong. On the other hand, in NELL such case was observed a few times, e.g. `ismultipleof(musicinstrument_herd , musicinstrument_buffalo)` whose object was corrected to `mammal_buffalo` but the subject remained wrong.

Also, our assumption that confusions tend to use a more general IRI instead of a more specific, requiring all tokens of the queried to be matched, does not always hold. One example in DBpedia which contradicts this assumption is `language(Paadatha_Thenikkal , Tamil_cinema)`, whose corrected object would be `Tamil_language` and could not be retrieved by our approach. While this can be a problem, dropping this assumption also means that more candidates entities will be retrieved, increasing the number of unrelated candidates, resulting in more candidate triples which need to be tested and possibly more occurrences of cases 2 and 3. Further experiments would have to be conducted in order to evaluate the effects of such change.

5 CONCLUSION

In this paper we proposed CoCKG, an approach for correcting erroneous facts originated from entity confusions. The experiments show that CoCKG is capable of correcting wrong triples with confused instances, with estimated precision of 21% of the produced corrections in DBpedia and 14% in NELL. The low precision values obtained do not allow this process, as of now, to be used for fully automatic KG enrichment. Nevertheless, it works as a proof of concept and can be useful, e.g., as suggestions from which a user would ultimately decide whether to execute.

In the future it would be interesting to adapt this method to support active learning. Since guaranteeing the quality of the newly generated facts is crucial, having input from the user to clarify borderline cases and improve the overall results would be highly valuable. Furthermore, using an ensemble of different KG models with different characteristics, e.g. KG embeddings, instead of a single model may potentially increase the robustness of the system. Finally, it would be worth adding textual features from entities descriptions to help determine if a pair of entities is related or not.

ACKNOWLEDGMENTS

The work presented in this paper has been partly supported by the Ministry of Science, Research and the Arts Baden-Württemberg in the project SyKo^{2}W^2 (Synthesis of Completion and Correction of Knowledge Graphs on the Web).

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. [n. d.]. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*.

[2] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo A. Baeza-Yates, and Sue B. Moon (Eds.). International World Wide Web Conferences Steering Committee / ACM, 413–422. http://dl.acm.org/citation.cfm?id=2488425

[3] Ni Lao and William W. Cohen. 2010. Relational Retrieval Using a Combination of Path-constrained Random Walks. *Mach. Learn.* 81, 1 (Oct. 2010), 53–67. https://doi.org/10.1007/s10994-010-5205-8

[4] Andre Melo and Heiko Paulheim. 2017. Detection of Relation Assertion Errors in Knowledge Graphs. In *Proceedings of the 9th International Conference on Knowledge Capture, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*. ACM.

[5] André Melo and Heiko Paulheim. 2017. Local and global feature selection for multilabel classification with binary relevance. *Artificial Intelligence Review* (2017), 1–28. https://doi.org/10.1007/s10462-017-9556-4

[6] André Melo, Heiko Paulheim, and Johanna Völker. 2016. Type Prediction in RDF Knowledge Bases Using Hierarchical Multilabel Classification. In *Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics (WIMS '16)*. ACM, New York, NY, USA, Article 14, 10 pages. https://doi.org/10.1145/2912845.2912861

[7] Stoyan Mihov and Klaus U. Schulz. 2004. Fast Approximate Search in Large Dictionaries. *Comput. Linguist.* 30, 4 (Dec. 2004), 451–477. https://doi.org/10.1162/0891201042544938

[8] Gonzalo Navarro. 2001. A Guided Tour to Approximate String Matching. *ACM Comput. Surv.* 33, 1 (March 2001), 31–88. https://doi.org/10.1145/375360.375365

[9] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33. https://doi.org/10.1109/JPROC.2015.2483592

[10] Maximilian Nickel, Volker Tresp, and Hans peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. ACM. http://www.icml-2011.org/papers/438_icmlpaper.pdf

[11] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* 8, 3 (2017), 489–508. https://doi.org/10.3233/SW-160218

[12] Heiko Paulheim and Christian Bizer. 2013. *Type Inference on Noisy RDF Data*. Springer Berlin Heidelberg, Berlin, Heidelberg, 510–525. https://doi.org/10.1007/978-3-642-41335-3_32

[13] Heiko Paulheim and Christian Bizer. 2014. Improving the Quality of Linked Data Using Statistical Distributions. *Int. J. Semant. Web Inf. Syst.* 10, 2 (April 2014), 63–86. https://doi.org/10.4018/ijswis.2014040104

[14] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. *CoRR* abs/1606.06357 (2016). http://arxiv.org/abs/1606.06357

[15] Chengyu Wang, Rong Zhang, Xiaofeng He, and Aoying Zhou. 2016. Error Link Detection and Correction in Wikipedia. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, New York, NY, USA, 307–316. https://doi.org/10.1145/2983323.2983705

[16] Zhenglu Yang, Jianjun Yu, and Masaru Kitsuregawa. 2010. Fast Algorithms for Top-k Approximate String Matching. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10)*. AAAI Press, 1467–1473. http://dl.acm.org/citation.cfm?id=2898607.2898841