Detecting and Correcting Typing Errors in DBpedia

Daniel Caminhas, Daniel Cones, Natalie Hervieux, and Denilson Barbosa

University of Alberta
Edmonton, AB, Canada
{caminhas,dcones,nhervieu,denilson}@ualberta.ca

Abstract. DBpedia has long been one of the major hubs of the Linked Open Data ecosystem. It is built by a largely automated process that uses many extractors and manually curated mappings to read information from infoboxes on Wikipedia. Given the complexity of the task, it is not surprising that DBpedia contains different kinds of errors, ranging from mistakes in the source text to errors in the extractors themselves (or in the order in which they are applied). Of particular importance are typing errors in which an entity is assigned a type from the DBpedia ontology to which it does not belong. These errors propagate very far, given the modern practice of relying on Knowledge Graphs (KGs) such as DBpedia for obtaining training data through distant supervision. We posit a way to correct these errors is through a *post factum* analysis of the KG. Thus, we introduce and evaluate a KG refinement approach that uses binary classifiers that rely on semantic embeddings of the entities to detect and remove incorrect type assignments. Our initial evaluation is done using a highly curated gold standard of 35 types from the DBpedia ontology and shows the method is very promising.

Keywords: knowledge graphs · entity embeddings · DBpedia.

1 Introduction

Knowledge Graphs (KGs) built from Web sources have been found effective for end-user applications such as question answering (e.g., YAGO [14] used in the IBM Watson System [4]) and data interlinking in the Linked Open Data (LOD) ecosystem. Moreover, such KGs are the primary source of training data for NLP applications following the distant supervision paradigm. Many approaches exist for building and maintaining KGs: they can be manually curated; collaboratively edited, like Freebase [1] and Wikidata [17]; or automatically extracted, like DBpedia [6]. Many companies have their own proprietary KGs, including Facebook, Google, Microsoft, and Diffbot, to mention a few.

Table 1. Number of inconsistencies on DBpedia identified using disjointness axioms.

Disjointness Axiom	**Number of Entities**
Place and Person	1761
Software and Company	1004
Person and University	1014
Person and Company	3892

Manually curated KGs tend to have high precision, often at the expense of coverage, while KGs automatically derived from Web sources have a high coverage but are susceptible to systematic errors. These errors may impact public image when manifested in user-facing applications such as web or social media search and can have far-reaching consequences as they propagate. Detecting and fixing these errors depends on the processes used and level of human involvement in creating the KGs.

Despite DBpedia's importance as a general use KG as well as its crucial role for the LOD movement, about 12% of DBpedia triples have some quality issues [19]. Most triples in DBpedia come from parsing the infoboxes of articles in Wikipedia. In principle, the infoboxes should follow strict templates with a list of attributes for the type of entity described in the article (e.g., person, organization, etc.). However, adherence to templates and editorial practices is hard to enforce, especially over time.

Typing Errors. One particularly problematic error in DBpedia concerns entity types. For example, at the time of writing, DBpedia says that the entity `dbr:Egypt` is a `dbo:MusicalArtist`. Similarly, `dbr:United_Nations` and `dbr:European_Union` are, among other things, also classified as a `dbo:Country`, together with another 7,106 entities, which seems unreasonably high, even accounting for entities that were historically identified as such. Table 1 shows other examples of type inconsistencies that can be identified using disjointness axioms [7]. Besides incorrect type assignments, DBpedia also suffers from the problem of *missing* types for some entities. For example, 27% of the 30935 entities classified as a `dbo:University` are not classified as an `dbo:Organization`.

We note that these errors, although problematic, are the exception instead of the norm in DBpedia. Moreover, we posit that, given the complexity and inherently noisy processes through which Wikipedia and DBpedia are created, the best way to correct these errors is through a *post factum* analysis of the entity types, which is what we seek to accomplish.

Throughout the paper, we use the customary `dbr:`, `dbo:`, and `dbp:` prefixes to indicate resources (which are entities), ontological predicates (e.g., types), and properties, respectively.

Wikipedia states that the United Nations have 193 members, while there are 8 other entities that are not members but are recognized as countries by at least one UN member.

Our Contribution. We propose the use of binary classifiers (one per type) to predict the type(s) of DBpedia entities. The classifiers rely on two kinds of semantic embeddings of the entities. From the Wikipedia text, we derive word2vec-like embeddings, while from DBpedia itself, we use PCA [16] to embed the entities based on their ontological properties. We test our method using a manually curated partial gold standard with 3876 entities of 35 different types of the DBpedia ontology. The performed experiments show that our approach is able to automatically find errors and assign types for DBpedia entities with over 97% accuracy.

2 Related Work

In the construction of a knowledge graph, there is a trade-off between coverage and correctness. To address those problems, some effort has been made to refine knowledge graphs. In contrast to the knowledge graph creation methods, the refinement techniques assume the existence of a knowledge graph which can be improved in a post-processing step by adding missing knowledge or identifying and removing errors [12].

One possible approach is to validate the knowledge graph manually using human annotators. Besides being costly, this approach is unfeasible for large databases such as DBpedia, due to its low scalability. Because of this, most researchers focus on developing automatic or semi-automatic solutions for knowledge graph refinement.

Many of the proposed solutions aim finding erroneous relations (i.e., the edges of the graph) between pairs of entities [2, 3, 5, 9, 13]. Meanwhile, others works aim to find incorrect literal values, such as numbers and dates. Identifying incorrect interlinks (links that connect entities representing the same concept in different graphs) between knowledge graphs has also been attempted [11]. A comprehensive survey on knowledge graph refinement methods is presented by Paulheim [12].

To the best of our knowledge, Ma *et al.* [7] was the first attempt at identifying incorrect type assertions. They proposed using disjointness axioms to detect inconsistencies. To create the axioms, they used association rule mining since it allows for the discovery of implicit knowledge in massive data. The axioms are learned from DBpedia and tested on DBpedia and Zhishi.me [10]. Although this approach is, in fact, able to identify several inconsistencies, it has a few limitations. First of all, the association rules are learned from DBpedia, which is itself a noisy dataset. Thus, there will always be some wrong axioms. Secondly, some entities on DBpedia are assigned to a single incorrect type. For example, the only assigned type for `dbr:Nail_polish` is `dbo:Person`, which is wrong. However, since there are no other types associated with this entity, there is no axiom capable of identifying this error, because each rule involves two classes.

In this work, we introduce resource2vec embeddings, which are vectors, similar to word embeddings [8], that represent entities on DBpedia. These embeddings are used as a feature for a set of machine learning classifiers that detect if the type assigned to an entity is correct. The intuition behind this approach is that embeddings of entities of the same type will be closer to one another in an n-dimensional continuous vector space than embeddings of entities of different types. For example, the similarity between two vectors for entities of the type Country (e.g., `dbr:Canada` and `dbr:United_States`) will be greater than the similarity between a vector of a country and a university (e.g., `dbr:Canada` and `dbr:Stanford_University`).

The usage of entity embedding for type detection on DBpedia was also proposed by Zhou *et al.* [20]. One important difference between our work and the one presented by Zhou *et al.* resides in the creation of the embedding. While they only use Wikipedia to train their embeddings, our embeddings are trained using properties from both Wikipedia and DBpedia. (as we explain in section 3). Another important difference is in the dataset used for training and testing. Zhou *et al.* uses DBpedia itself to create the datasets. They query a public DBpedia SPARQL endpoint to select, for each DBpedia type, entities as positive examples of that type. Negative examples are chosen from a random selection of instances from all the remaining types. We argue that this approach will create a noisy dataset since, as we discussed, many entities on DBpedia have incorrectly assigned types, and that is exactly the problem that we are attempting to solve. In this work, we use a manually curated partial gold standard for training and testing.

3 Method

3.1 Representing DBpedia Entities

Our approach consists of creating a semantic mapping of DBpedia resources, which is used as a feature for a set of binary machine learning classifiers. For that, we concatenate wikipedia2vec and DBpedia2vec embeddings. The wikipedia2vec are word2vec-like embeddings that represent Wikipedia entities. They are created using Wikipedia2Vec [18], a tool that allows learning embeddings of words and entities simultaneously, and places similar words and entities close to one another in a continuous vector space.

DBpedia2vec are embeddings that help to represent a DBpedia entity. They are created using the entity's properties (i.e., predicates in the RDF tuples) on DBpedia. Our intuition is that most entities of the same type share the same properties. For example, countries usually have properties such as `dbo:areaTotal`, `dbo:capital`, and `dbo:largestCity`, while people are more likely to have properties like `dbo:birthDate`, `dbo:birthPlace`, and `dbo:nationality`.

To create DBpedia2vec, we create a list of all distinct properties existing in DBpedia (ignoring properties that are common across all types on DBpedia ontology, such as `dbo:wikiPageID`, `dbo:wikiPageWikiLink`, `dbo:abstract`, and `dbo:sameAs`). A one-hot encoding vector is created for the entity. Each dimension of this vector represents one of the 3480 properties of DBpedia. Then, we apply a probabilistic principal component analysis (PCA) [16] to linearly reduce the dimensionality of the embeddings using Singular Value Decomposition of the data. In this way, we are able to project the 3480-dimension embeddings to a lower dimensional and continuous space with $n_2 = 300$ dimensions.

3.2 Identifying and correcting erroneous types

The resource2vec embeddings are used as a feature by a binary classifier which is trained to determine if the type assigned to a resource is correct. One classifier is trained for each type, using resource2vec embeddings of resources that belong to that type as positive examples and resource2vec embeddings of randomly selected resources from all other types as negative examples.

This approach allows us to not only identify erroneous type assignments but also to assign the correct type to any DBpedia resource for which the resource2vec embedding is created, even if no type has been assigned yet on DBpedia. We tested the classification using three algorithms: Naive Bayes, K-nearest neighbours (K-NN), and nearest centroids, which represents each class (i.e., each type) by its centroid and assigns the class of the nearest centroid to test samples [15].

4 Experiment setup and Results

The experiments were performed using resource2vec embeddings created by concatenating 500-dimensional wikipedia2vec embeddings trained on a Wikipedia dump extracted Feb. 2019 and 300-dimensional dbpedia2vec trained on the 2016 release of DBpedia. In an attempt to obtain high-quality embeddings, the wikipedia2vec embeddings were trained with 10 iterations over the articles, a windows size of 10, and a minimum number of 10 occurrences for words and 5 occurrences for entities.

Gold Standard. To better evaluate our classifiers, we created a gold standard encompassing the following 35 types from the DBpedia ontology: *Aircraft, Airline, Airport, Album, AmericanFootballPlayer, Animal, Automobile, Bacteria, Bank, Book, Building, City, Country, Currency, Food, Galaxy, HorseTrainer, Language, MilitaryConflict, Murderer, MusicalArtist, MythologicalFigure, Planet, Plant, President, Software, Song , Sport, Swimmer, Theatre, TimePeriod, Train, University, Volcano,* and *Weapon.* We chose these types with the goal of maximizing the diversity of entities while minimizing inter-type overlap (which could potentially confuse our analysis and preliminary conclusions). If the approach works well in this simplified setting, it may be worth scaling the solution to consider all types on DBpedia.

To build the gold standard, annotators were asked to use any resources at their disposal (e.g., Wikipedia's own entity lists or categories) to find examples of entities in each of the 35 types. In total, we selected 3876 entities. The number of entities per type varied from 94 (for the types *Sport* and *Software*) to 112 (for

Table 2. Comparison between the algorithms used for creating binary classifiers for error detection on type assignment.

Classifier	**P**	**R**	**F1**	**Accuracy**
Nearest Centroid	**0.98**	**0.97**	**0.97**	**0.97**
k-NN	0.95	0.95	0.95	0.95
Naive Bayes	0.89	0.89	0.89	0.89

Table 3. Estimated performance of Nearest Centroid classifiers on unseen entity-type pairs.

Prediction	**Number of predictions**	**Manually verified samples**	**Accuracy**
Correct	251756	2448	0.96
Incorrect	113035	1251	0.76

the type *Aircraft*). All of our testing and evaluation data can be downloaded from *https://bit.ly/2FcqQQW*.

The Need For Manual Annotations. An alternative to the manual annotation and evaluation that we followed here would be exploiting an independent KG (e.g., Google's knowledge graph) for the evaluation. In principle, such an external KG could be used to correct typing errors on its own. We attempted such an approach but ran into several difficulties. First, there is the issue of the incompleteness of the external KG itself. We found that generally only high-level types are assigned to entities in the Google KG: for instance, most entities of type *Aircraft* in DBpedia are labeled simply as *Thing* in the Google KG. Second, DBpedia interlinks to other KGs are wrong or missing up to 20% of the time [19], therefore, finding equivalent entities on different knowledge graphs is a challenging task itself. Finally, the ontology of the KGs may be significantly different, for example, we noticed that other KGs (e.g., YAGO), has significantly more types than DBpedia.

4.1 Comparing classifiers

Our first experiment consisted of comparing popular binary classifiers suitable for the task. We used 70% of the entities of the gold standard for training the classifiers and the remaining 30% for testing them. Hyperparameter tuning for k-NN was performed using 5-fold cross-validation. Table 2 shows the results. The reported values are an average of 10 runs on different training/testing splits of the gold standard. For each run, we averaged the precision, recall, F1-Score, and accuracy of the 35 binary classifiers. The Nearest Centroid approach leads to better classifiers, achieving more than 97% of performance in all metrics, while the Naive Bayes classifiers achieved the lowest performance.

4.2 Predicting types of unseen entities

Motivated by the high accuracy of the binary classifiers, we tested whether the proposed supervised approach could detect incorrect type assignments among other entities of the 35 classes in our gold standard. For this, we used the best performing algorithm (Nearest Centroid). In total, 364,791 entity-type pairs were checked by the classifier: a positive classification confirmed the type prediction while a negative classification disproved it. Human annotators verified the output of the classifiers for a random sample of 3699 predictions. Table 3 shows the results.

Upon further inspection of the false negatives, we noticed some discrepancies in the way entities are classified in DBpedia which were not reflected in the way we created our gold standard. The most notable example concerns the class `dbo:Animal`, for which most instances correspond to Wikipedia articles describing a *species* (e.g., `dbr:American_black_bear`). In fact, all instances in our gold standard correspond to species. In our test sample, however, we found many *individual* racehorses also classified as `dbo:Animal` (e.g., `dbr:Fusaichi_Pegasus`). Not surprisingly, all such instance-type pairs were (correctly in our opinion) rejected by our classifier. To further illustrate our claim, we note that racehorses have properties like `dbo:honours`, `dbo:owner`, `dbo:sex`, `dbo:trainer`, and `dbp:earnings`, while most other instances with `dbo:animal` type have properties such as `dbo:family`, `dbo:genus`, `dbo:kingdom`, `dbo:order`, `dbo:phylum`, and `dbo:conservationStatus`.

We found other similar cases involving other ontology types. In order to more accurately evaluate the effectiveness of the classifier, we *removed* from our analysis the following cases:

– Animals that are also an instance of type racehorse.
– Cities that are fictional or medieval cities.
– Automobiles that are buses or trucks.
– Songs that are rhymes, prayers, hymns, lullabies, or marches.
– Countries that are fictional countries, former sovereign states, or former kingdoms.

The manual inspection showed that the proposed approach has a very low false positive rate of less than 5%, which is very encouraging. Moreover, the method is correct about 75% of the times it claims a type assignment is wrong, for a false negative rate below 25%. The performance of the classifier varies across classes: for example, both false positive and false negative rates for entities tagged as `dbo:HorseTrainer` is 0%. On the other hand, the false positive rate for the class `dbo:President` is 13%. That is probably because `dbp:President` is a more generic class, which can include presidents of countries, universities, companies, institutes, associations, councils, etc. This could be addressed by increasing the diversity of entities in the training data.

Some entities had multiple types.

Table 4. Classification results using embeddings created from YAGO types

Classifier	P	R	F1	Accuracy
Nearest Centroids	**0.94**	**0.93**	**0.93**	**0.93**
K-NN	0.90	0.89	0.89	0.89
Naive Bayes	0.88	0.84	0.84	0.84

4.3 Predicting with YAGO types

Since DBpedia entities are annotated with types from YAGO, we also attempted to leverage these links for identifying and correcting typing errors in DBpedia. However, these two ontologies cannot be easily aligned: we found 419,297 unique objects with a YAGO prefix for which there was a predicate `rdf:type` associated with a DBpedia entity. Thus, our attempt to use YAGO boiled down to: for each DBpedia entity *in our gold standard*, we created a one-hot encoding vector that represents the YAGO types assigned to that entity, then we apply PCA to reduce the dimensionality of the one-hot encoding vectors. From those vectors, we created binary classifiers as in Section 4.1. The results are shown in Table 4.

Although the embeddings created using all YAGO types seem to carry a strong signal, it is clear we need a way to filter out a large number of types before we can obtain embeddings for entities other than in the gold standard. Furthermore, we observed several entities in the gold standard with identical sets of YAGO types, which means that the classifiers may be overfitting, rendering the numbers in Table 4 unreliable.

5 Conclusion and Future Work

This paper presented an effective approach for detecting erroneous type assignments in a KG by leveraging an annotated corpus of text and the properties in the KG itself. The assumptions behind the method are that the input KG is of sufficient quality so that the initial type assignment is not random. Moreover, our method can be applied *post factum*, without requiring any changes to the already complex KG generation process. We evaluated the approach on a large and carefully created gold standard, and obtained very encouraging results. We used the best performing binary classifiers to verify the type assignment for thousands of unseen entity-type pairs and found out that the false positive rate of the method is below 5%, while the false negative rate is below 25%. We believe these results are encouraging. The method also found some inconsistencies in the way types are assigned to entities.

There are several interesting directions for future work. First, the method described here is supervised, and although expanding our gold standard to en-

We were not able to obtain embeddings for all entities among the 35 types, let alone embeddings for all entities in DBpedia, due to the time complexity of PCA analysis and the size of the input matrix.

compass the 537 classes in DBpedia certainly seems within reach of its community, we seek to develop a fully unsupervised method for selecting representative entities for each class to be used to derive the centroids. Another interesting idea would be to perform a detailed assessment of the DBpedia ontology and try to identify types that are too broad and should be split into multiple subtypes, and types that are too specific and could be merged with others. Also, we believe it would be interesting to evaluate our method for the task of identifying missing types (as opposed to incorrect ones). Finally, although we tested on DBpedia only, we believe our method could be easily adapted to find errors on other knowledge graphs provided one can find an annotated text corpus.

Acknowledgements

This work was done with the support of the Natural Sciences and Engineering Research Council of Canada and gifts from NVIDIA and Diffbot Inc.

References

1. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.
2. Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM, 2010.
3. Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.
4. David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.
5. Jens Lehmann, Daniel Gerber, Mohamed Morsey, and Axel-Cyrille Ngonga Ngomo. Defacto-deep fact validation. In *International semantic web conference*, pages 312–327. Springer, 2012.
6. Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.
7. Yanfang Ma, Huan Gao, Tianxing Wu, and Guilin Qi. Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data. In *Chinese Semantic Web and Web Science Conference*, pages 29–41. Springer, 2014.

8. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
9. Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 227–236. ACM, 2011.
10. Xing Niu, Xinruo Sun, Haofen Wang, Shu Rong, Guilin Qi, and Yong Yu. Zhishi. me-weaving chinese linking open data. In *International Semantic Web Conference*, pages 205–220. Springer, 2011.
11. Heiko Paulheim. Identifying wrong links between datasets by multi-dimensional outlier detection. In *WoDOOM*, pages 27–38, 2014.
12. Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.
13. Heiko Paulheim and Christian Bizer. Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):63–86, 2014.
14. Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.
15. Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, 2002.
16. Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.
17. Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledge base. 2014.
18. Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Wikipedia2vec: An optimized tool for learning embeddings of words and entities from wikipedia. *arXiv preprint 1812.06280*, 2018.
19. Amrapali Zaveri, Dimitris Kontokostas, Mohamed A Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 97–104. ACM, 2013.
20. Hanqing Zhou, Amal Zouaq, and Diana Inkpen. Dbpedia entity type detection using entity embeddings and n-gram models. In *International Conference on Knowledge Engineering and the Semantic Web*, pages 309–322. Springer, 2017.