Identifying Wrong Links between Datasets by Multi-dimensional Outlier Detection

Heiko Paulheim

University of Mannheim, Germany
Research Group Data and Web Science
heiko@informatik.uni-mannheim.de

Abstract. Links between datasets are an essential ingredient of Linked Open Data. Since the manual creation of links is expensive at large-scale, link sets are often created using heuristics, which may lead to errors. In this paper, we propose an unsupervised approach for finding erroneous links. We represent each link as a feature vector in a higher dimensional vector space, and find wrong links by means of different multi-dimensional outlier detection methods. We show how the approach can be implemented in the RapidMiner platform using only off-the-shelf components, and present a first evaluation with real-world datasets from the Linked Open Data cloud showing promising results, with an F-measure of up to 0.54, and an area under the ROC curve of up to 0.86.

Keywords: Linked Open Data, Link Quality, Data Quality, Link Debugging, Outlier Detection

1 Introduction

Links between datasets are an essential ingredient for Linked Open Data [6]. For reasons of scalability, such interlinks are often not created manually, but generated (semi-)automatically by heuristics, which leads to occasional wrong links.

There are different reasons why link sets may contain errors. The first (and probably most frequent) reason is that the **heuristic mechanism** that creates the links does not work at an accuracy of 100%. Typical heuristic approaches for generating links combine different string metric of the entities' labels, sometimes combined with some filtering by type (e.g., only linking entities of type *Person*) [26]. Those heuristics can work well, but are not free from errors, e.g., linking two different persons which share the same name, or a river and a region with the same name. Moreover, with such heuristics, there is a trade-off between recall and precision, which leads to incorrect links. For example, [28] reports that around 20% of the links between DBpedia and Freebase are incorrect. A further problem is that the link generation heuristics are usually not re-created every time one of the linked data sources changes, thus, links may be outdated, e.g., pointing to to resources that do not exist anymore.

Another source of errors is that entities are linked which are **not exactly the same**. While in theory, entities linked by `owl:sameAs` should refer to the same real-world entity, this is not often the case, e.g., when linking a description of the company *Starbucks* to an actual Starbucks café. A study in 2010 has shown that only about half of all `owl:sameAs` actually denote two descriptions of the same real world entity [13].

In order to increase the quality of links between datasets, we propose an approach which uses multi-dimensional outlier techniques for detecting wrong links. To that end, features for each link are created, so that the link can be described as a point in a high dimensional feature space. We use outlier detection methods to find those links that are represented by points which are far from the overall distribution, assuming that those points represent wrong links.

The rest of this paper is structured as follows. In section 2, we show our approach for finding wrong links with outlier detection. In section 3, we introduce the experimental setup we used for validating our approach, and discuss the results. We conclude the paper with a review of related work in section 4, and an outlook on future work in section 5.

2 Approach

For finding wrong links with outlier detection, we first represent each link as a feature vector. Possible features are, e.g., the direct types of resources in the linked datasets, i.e., all objects of statements that have the linked resource as a subject and `rdf:type` as a predicate. A simplified example is shown in Fig. 1: two datasets contain links between artists and music works. Instances of `Song` and `Album` in dataset 1 are linked to instances of `Music Work` in dataset 2, and instances of `Artist` in dataset 1 are mapped to instances of `Music Artist` in dataset 2. It can be observed that in that feature space, there are relatively dense clusters, and single outliers (such as the one dot in the upper middle, which represents an album wrongly linked to an artist). Assuming that the majority of links between two datasets is correct, the clusters are likely to represent the correct links, while the singular outliers are likely to be wrong links.

Such singular outliers can be found by methods of *outlier* or *anomaly detection* [8, 15]. These methods automatically assign labels or scores to data points which significantly deviate from the majority of data points in the overall dataset. The outlier detection approach to be used has to be *multi-dimensional*, i.e., find data points that are abnormal w.r.t. the combination of their coordinates. In contrast, *single-dimensional* or *univariate* outlier detection methods (such as Grubbs' test or IQR) find suspicious data points in only one dimension, e.g., unusually large or small temperature values measured by a sensor. In Fig. 1, the outlying data point would not be an outlier if only considering one dimension, i.e., only the type in dataset 1 or the type in dataset 2.

To facilitate the detection of wrong links by outlier detection, our approach consists of three basic steps:

1. Read a link set, and create a feature vector representation for each link

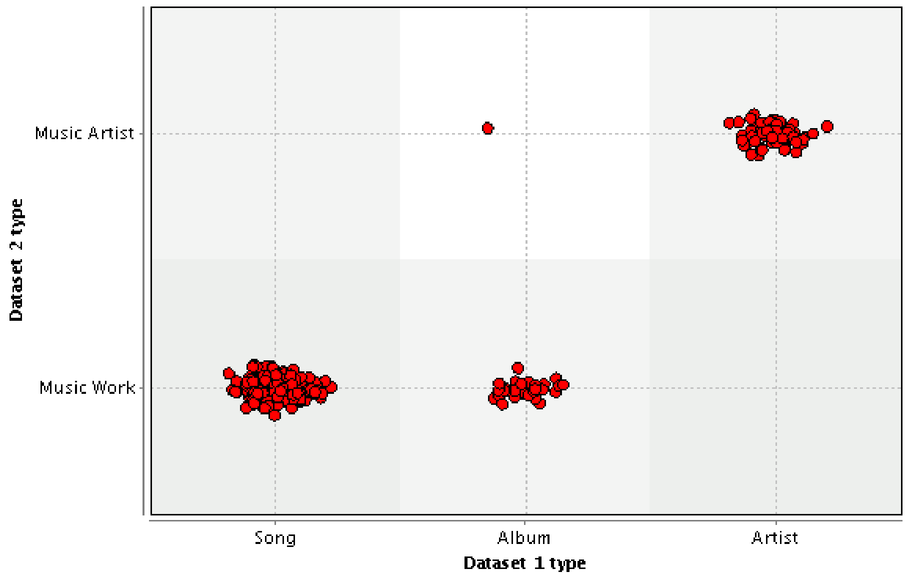

Fig. 1. A simplified example of links being represented in a vector space. The single dot in the upper middle quadrant represents a wrong link.

2. Perform outlier detection on the set of vectors, i.e., assign an outlier score to each link
3. Order the links by outlier score, and store them

In a semi-automatic setting, a user would work through the list from top to bottom until the false positive rate begins to rise above a certain limit. For fully automatic link correction, all links with an outlier score above a threshold τ would be regarded as outliers.

3 Experiments

To evaluate our approach, we have set up a process in the RapidMiner[1] platform for data mining, combining operators from the Linked Open Data extension [23] and the Anomaly Detection extension [11]. The basic pipeline is shown in Fig. 2: first, a set of links is read, e.g., from a SPARQL endpoint, and for both resources linked, features are added to the feature vector representation using the Linked Open Data extension. The resulting vector is then passed to an outlier detection algorithm, which assigns outlier scores. The output is written to a file containing pairs of resources, augmented with scores.[2]

[1] `http://www.rapidminer.com`

[2] A step-by-step explanation of how to set up such a process is shown at `http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension/rapidminer-lod-extension-example-discovering-wrong-links-between-datasets/`

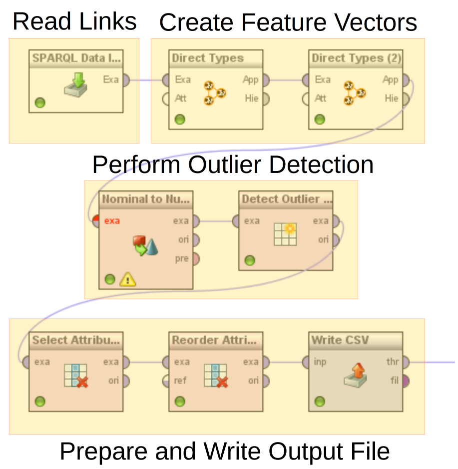

Fig. 2. The implementation of our approach in the RapidMiner platform

3.1 Feature Vector Creation

We examine two different strategies of creating feature vectors:

– *Using all direct types.* A binary feature is created for each schema class, which is set to `true` for a link if the linked resource has the class defined as its `rdf:type`.
– *Using all ingoing and outgoing properties.* Two binary features are created for each data and object property, which are set to `true` if the linked resource is the subject resp. the object of a triple which uses the property as its predicate.

The same feature creation technique is applied to each of the two linked resources, where distinct features are created for both resources. Furthermore, we examine the union of both feature sets.

3.2 Datasets

We evaluate our approach on two link sets between three datasets of the Linked Open Data cloud. The three datasets are:

– *DBpedia*, a cross-domain dataset created from Wikipedia infoboxes [20].
– *Peel Sessions*, a dataset describing the John Peel Sessions at BBC, the artists involved, and the songs performed [25].
– *DBTropes*, a dataset collecting information about movies, TV shows, computer games, and books, among others, as well as tropes used in those [17].

For DBpedia, we use the mapping-based types and the mapping-based properties datasets of the 3.9 release[3]. For the Peel Sessions dataset, we use the dump

[3] `http://wiki.dbpedia.org/Downloads39`

Table 1. Sizes of feature vectors for the different link sets

Dataset	Peel Session	DBpedia	DBTropes	DBpedia
# Links	2,087		4,229	
# Types	3	31	2	79
# Properties	4	56	18	124

available at the web site[4]. For the DBTropes dataset, which provides daily snapshots, we use a snapshot obtained on November 8th, 2013.[5]

Both the Peel Sessions and the DBTropes data set are linked to DBpedia. The Peel Sessions dataset contains 2,087 `owl:sameAs` links to DBpedia, the DBTropes dataset contains 4,229 `owl:sameAs` links to DBpedia. While the Peel Sessions dataset is rather restricted to the type of entities it links (in particular: artists and songs), DBTropes contains a larger variety of entities, including general concepts such as *Celtic Mythology*.

Besides random links two homonymous resources (e.g., the TV series *Material Girl* and the Madonna song), one typical source of errors is the linking of instances derived from disambiguation pages (both DBpedia and DBTropes, which is also derived from a Wiki, have such instances). A typical source of errors for the Peel Session dataset is the linking of songs to albums with the same name. Furthermore, the Peel Sessions dataset links different persons of the same name – e.g., a blues musician named *Jimmy Carter* to the U.S. president.

Table 1 depicts the sizes of the feature vectors for both link sets, i.e., the number of classes and properties used for the elements that are mapped. The counts of DBpedia classes and properties, show that the variety of objects linked from DBTropes is higher. Furthermore, it is noteworthy that although DBTropes uses two classes, one of those is only used for two objects, while the remaining 4,219 instances have the class `TVTItem`, which is only a generic class comparable to `owl:Thing`. The properties used in the dataset are similarly generic.

For our experiment, we have randomly sampled 100 links from both link sets, and manually evaluated them for correctness, thus creating small partial gold standards. From the Peel Session link set, 90 out of the 100 links are correct, for the DBTropes link set, 76 out of the 100 links are correct. For the gold standard, we use a strict definition of `owl:sameAs`, e.g., a book and its protagonist are not considered the same, neither are a book and a movie based on that book.

3.3 Outlier Detection Methods

To detect outliers, we compare six different multi-dimensional outlier detection methods. For all outlier detection methods, we use the implementation in the RapidMiner Anomaly Detection extension, and used the default parameters unless specified otherwise.

– The *k-NN global anomaly score (GAS)* is the average distance to the k nearest neighbors [5], following the intuition that outliers are located in rather

[4] `http://dbtune.org/bbc/peel/`, downloaded on November 6th, 2013

[5] `http://skipforward.opendfki.de/wiki/DBTropes`

sparsely populated areas of the vector space (cf. Fig. 1). Since values for k between 10 and 50 are recommended [11], we compute a GAS with $k = 10$, $k = 25$, and $k = 50$.

- The *Local Outlier Factor (LOF)* is computed from the density of data points around the point under inspection, which in turn is computed from the distances to the k nearest neighbors [7]. Since the algorithm allows the setting of a minimum and a maximum k, we use $k_{min} = 10$ and $k_{max} = 50$ following the recommendation above.
- The *Local Outlier Probability (LoOP)* follows a similar idea as LOF, but maps the outlier scores to probabilities in a $[0; 1]$ interval (the scores assigned by other methods are usually unbound) [19]. Like for GAS, we compute LoOP with $k = 10$, $k = 25$, and $k = 50$.
- The *Cluster-based Local Outlier Factor (CBLOF)* uses the output of a clustering algorithm. It follows the intuition that outliers are located outside of larger clusters, and thus assigns an outlier score based on the size of the cluster in which a data point is located, and the distance to the next large cluster [14]. According to the recommendation in [11], we set the α value to the expected percentage of correct instances, i.e., 0.90 for the Peel dataset, and 0.76 for the DBtropes dataset.[6] As a clustering algorithm, we use the *X-means* algorithm, which restarts k-means with different values for k, in order to find an optimal one [24]. For the X-means clustering, we set $k_{min} = 2$ and $k_{max} = 60$.
- The *Local Density Cluster-based Outlier Factor (LDCOF)* works similar to CBLOF, but also takes the local density of the cluster into account [3]. We again use it together with X-means in the same configuration as above.
- *One-Class Support Vector Machines* aim at training a support vector machine covering only positive examples, so that the majority of data points is separated from the rest. In our experiment, we use one-class SVMs with a robust kernel defined particularly for outlier detection [4].

Most of the above methods (including the clustering algorithm) require the definition of a distance function. Here, we use *cosine similarity*, since we want two links to be more similar if they share a feature (both are of type `Person`), but not if they share the absence of a feature (e.g., both are *not* of type `City`). In contrast, other distance functions, such as Euclidean distance, would weigh the shared presence and absence of a feature equally.

3.4 Results

We have tested each of the above outlier detection methods with three different feature groups – direct types, properties, and the combination of both – on both datasets, performing a total of 60 runs of the approach. The results are depicted in table 2. We report the area under the ROC curve (AUC), the best F1-measure

[6] Strictly speaking, setting these values according to observations on a labeled sample of the data makes the approach using CBLOF no longer fully supervised.

Table 2. Results on both datasets using different feature sets and methods. For each dataset, the top three AUC and F-measure values are marked in bold.

Dataset	Peel				DBTropes			
Features / Method	AUC	F1	τ	total	AUC	F1	τ	total
types								
GAS (k=10)	0.353	0.185	1.414	2,049	0.404	0.390	0.000	4,088
GAS (k=25)	0.341	0.182	0.476	2,071	0.424	0.390	0.000	4,009
GAS (k=50)	0.341	0.182	0.478	2,071	0.422	0.390	0.000	3,943
LOF	0.753	0.454	0.953	1,843	**0.619**	**0.500**	1.084	3,025
LoOP (k=10)	0.749	0.454	0.311	1,834	0.413	0.412	0.000	1,636
LoOP (k=25)	**0.803**	**0.500**	0.378	1,181	**0.581**	**0.488**	0.143	2,978
LoOP (k=50)	**0.803**	**0.500**	0.378	1,181	**0.581**	**0.488**	0.920	2,969
CBLOF	0.754	**0.537**	245.423	1,051	0.413	0.404	0.000	1,498
LDCOF	0.696	0.432	0.953	1,352	0.410	0.404	0.000	1,498
1-class SVM	**0.857**	0.471	2.689	1,514	0.456	0.421	3.712	1,795
properties								
GAS (k=10)	0.341	0.182	0.955	2,059	0.411	0.387	0.000	786
GAS (k=25)	0.344	0.182	0.969	2,046	0.405	0.387	0.000	563
GAS (k=50)	0.381	0.182	0.000	663	0.391	0.387	0.000	461
LOF	0.516	0.217	1.102	1,225	0.529	0.424	0.984	1,006
LoOP (k=10)	0.364	0.222	0.156	1,810	0.510	0.425	0.076	2,037
LoOP (k=25)	0.438	0.250	0.706	1,992	0.422	0.387	0.000	1,060
LoOP (k=50)	0.452	0.235	0.531	1,966	0.489	0.411	0.000	1,012
CBLOF	0.402	0.189	68.426	426	0.496	0.400	197.739	254
LDCOF	0.516	0.208	1.013	1,509	0.428	0.390	0.619	276
1-class SVM	0.360	0.189	2.000	426	0.378	0.387	2.000	200
all								
GAS (k=10)	0.331	0.200	0.553	1,942	0.412	0.387	0.000	785
GAS (k=25)	0.349	0.200	0.591	1,927	0.407	0.387	0.000	562
GAS (k=50)	0.440	0.222	0.520	1,529	0.390	0.387	0.000	460
LOF	0.638	0.280	1.105	1,002	0.481	0.400	1.010	567
LoOP (k=10)	0.454	0.333	0.802	2,063	0.547	0.420	0.064	1,881
LoOP (k=25)	0.430	0.250	0.478	2,004	0.445	0.388	0.000	1,065
LoOP (k=50)	0.378	0.235	0.473	1,980	0.502	0.420	0.008	1,253
CBLOF	0.313	0.189	25.302	235	0.366	0.403	223.036	240
LDCOF	0.530	0.250	1.326	1,876	0.467	0.390	0.632	272
1-class SVM	0.303	0.180	2.000	237	0.353	0.387	2.000	199

that can be achieved, the threshold τ that has to be set on the outlier score in order to achieve that F1-measure, and the total number of outliers that are identified at that threshold.

Multiple observations can be made from the table. First, in particular in terms of AUC, the results on the Peel dataset are much better than those on the DBTropes dataset. There are two main reasons for that: on the one hand, the schema used in the Peel dataset is more fine-grained than that of the DBTropes dataset, where the latter essentially has only major class, which is `TVTItem`. Second, with around 24%, the fraction of outliers on the DBTropes dataset is rather large, and larger than the amount of outliers many outlier detection methods are built for. This can be observed very well on the results for the 1-class SVM method, which reaches the best AUC on the Peel dataset, but performs only average on the DBTropes dataset.

Second, using only the type features works best, and the results do not improve when combining both feature sets. As shown in table 1, the number of features created from direct types is much smaller than that created from relations, i.e., the outlier detection problem to be solved has a much lower dimensionality. A large number of dimensions, however, is a problem for many outlier detection methods, in particular those based on nearest neighbor methods. The combination of type features and LoOP yields good results, with an AUC of 0.803 and 0.581, respectively, while the optimal results are achieved by the 1-class SVM (AUC=0.857) and CBLOF (F1=0.537) on the Peel dataset, and by LOF (AUC=0.619, F1=0.5) on the DBTropes dataset. The absolute numbers of identified outliers for the optimal F1 show that in those cases, the F1 is optimized mainly because of a high recall value, flagging up to three quarters of all links as outliers. This shows that selecting an optimal configuration is difficult.

In order to obtain a more fine-grained picture of the differences between the approaches, figures 3 and 4 show the ROC curves of all approaches, using only type features. It shows that in particular the LoOP approaches show very good results on both datasets. The steep ascend of the respective ROC curves show that there are five actually wrong links among the top 10 identified outliers.

The approach runs very fast in most cases. While the creation of feature vectors strongly depends on the data access method (e.g., working with a public SPARQL endpoint over the internet is much slower than using a local dump), the outlier detection itself takes less than 10 seconds on both datasets for all the methods used in our experiments. The only exceptions are the clustering-based methods, where the clustering can take up to 30 seconds, and most dominantly the One-Class SVM method, which can take up to 15 minutes.

4 Related Work

In this paper, we have analyzed the use of multi-dimensional outlier detection for finding erroneous links. This work is orthogonal to the approach sketched in [27], where we use outlier detection in a one-dimensional setting to find wrong numeric literals in DBpedia.

While a larger body of work is concerned with automatically *creating* links, there are not too many approaches that try to automatically find errors in links between datasets. Moreover, most approaches discussed so far assume some prior knowledge about the datasets, e.g., links on the schema level.

[12] use a set of five network metrics, such as degree and centrality, to predict typical properties of nodes in two interlinked datasets, as well as try to find wrongly linked resources. They report a recall of 0.68 and a precision of 0.49 (although on a different dataset), i.e., a result quality comparable to the approach discussed in this paper. In [9], links between more than two datasets are exploited to find the set of `owl:sameAs` that minimize the contradictions. The authors

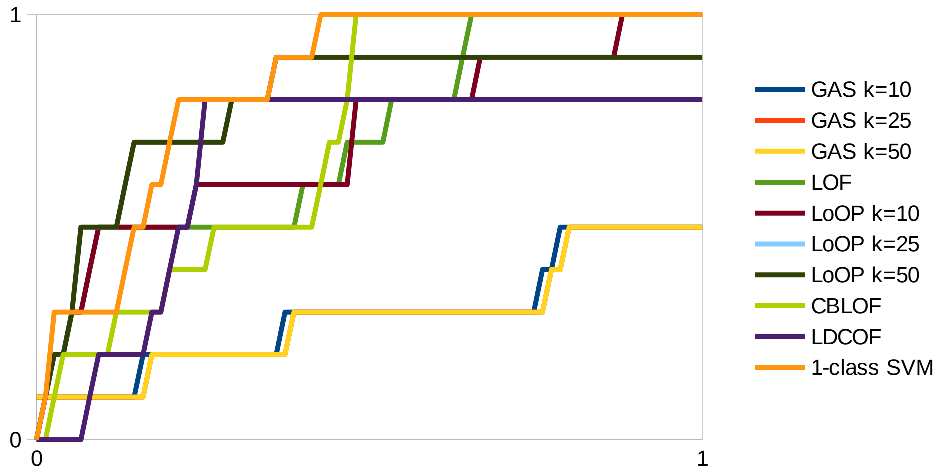

Fig. 3. ROC curve of the results on the Peel dataset, using only type features. The three curves for GAS are mostly identical; so are LoOP for $k = 25$ and $k = 50$.

show that they are capable of identifying a significant amount of contradictions, however, they do not state the precision of their approach. A similar problem is addressed in [10], where the authors aim at finding the most coherent set of links from a set of possible link candidates.

An approach using statistical distributions of properties, such as average degrees, is discussed in [16]. Like our approach, the authors compute confidence scores for `owl:sameAs` links. However, there is a fundamental difference: the authors expect the same schema to be used by both linked resources. In contrast, our approach can cope with entities using different schemas. The two link sets used in this paper could not have been processed with such an approach expecting the same schema for both linked datasets.

The *Databugger* framework allows for finding typical patterns of wrong and/or incomplete data, formulated as SPARQL queries [18]. The key difference is that, while Databugger relies on schema information (e.g., `owl:equivalentClass` definitions), our approach is agnostic with respect to the schemas used in the datasets at hand. In [1], a crowd sourcing approach is introduced for evaluating the quality of interlinks between datasets. While a considerable precision of 0.94 is achieved using majority voting over Amazon MTurk tasks, the results are not directly comparable, since the approach discussed in this paper works fully automatically and unsupervised, while the authors exploit the wisdom of the crowd. In [2], an approach is discussed for assessing the completeness of link sets, based on manually defined schema mappings. This is complementary to our work, which is concerned with *correctness*, not *completeness*.

The approaches in this paper focus on debugging link sets between individuals, i.e., links on the A-box level. A related problem is the debugging of schema mappings, i.e., links on the T-box level. Here, reasoning based approaches are frequently used [21]. While reasoning would also be a possible approach for A-box level link set debugging, the problems here are scalability and missing expressiv-

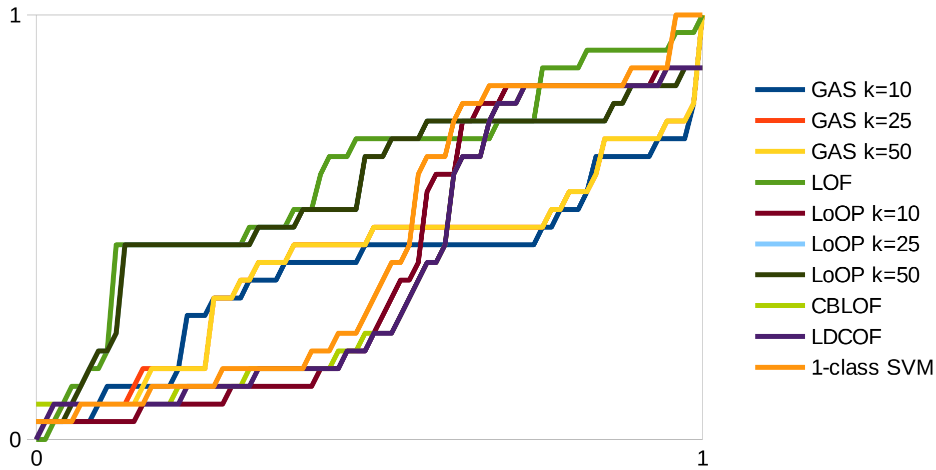

Fig. 4. ROC curve of the results on the DBTropes dataset, using only type features. The curves for GAS $k = 25$ and $k = 50$ are mostly identical; so are the curves for LoOP with $k = 25$ and $k = 50$, and the curves for CBLOF and LDCOF.

ity of the schemas used for Linked Open Data, and the A-box data often being too noisy for reasoning to yield useful results [22].

5 Conclusion and Outlook

In this paper, we have presented an approach for finding wrong links between datasets, which uses multi-dimensional outlier detection techniques. An evaluation on two datasets has shown promising results, with an area under the ROC curve up to 0.86 (i.e., wrong links get lower scores than correct links with a probability of 86%), and an F-measure up to 0.54. The approach is scalable, as it processes link sets between real datasets from the LOD cloud in a few seconds to a few minutes, depending on the configuration used.

Although the datasets used for evaluation only use `owl:sameAs` links, it can be applied to all sorts of datasets interlinks, the approach is not limited to a particular type of links. It may also be used, e.g., on a dataset of persons linked to a dataset of locations using `foaf:basedNear` links, or even for finding wrong instantiations of any property *within* a single dataset.

Given the amount of work that has been done in supervised or active learning of dataset interlinks, a link validation method such as the one introduced in this paper could be an interesting counterpart to be used in such learning systems. Given that the features used for learning and for validating the links are different, our method could provide a direct feedback loop for refining the learned links.

In essence, there are two basic degrees of freedom in our approach: the strategy for creating feature vectors, and the outlier detection algorithm (and its parametrization). With respect to feature vectors, we have experimented with direct types and properties so far. A further option are qualified relations, as discussed in [23], which, however, may impose scalability issues. Network measures, as discussed in some related works, are an interesting option for generating possible features, and domain or dataset specific features, such as Wikipedia categories for DBpedia, may also be csonidered. Furthermore, since many outlier detection algorithms experience problems in higher dimensional spaces, applying feature selection might be a useful preprocessing step, which, however, has to be taken with great care, since particularly in our setting, the very sparse features (which are likely to be eliminated by many feature selection approaches) are often those which are well suited for finding outliers.

As far as the selection of outlier detection methods is concerned, we have observed some trends, in particular that Local Outlier Factor, Local Outlier Probabilities, and 1-class SVMs perform quite well, however, especially the latter two need to be carefully parametrized. Since many automatic parameter tuning methods rely on a supervised rather than an unsupervised setting, it might be an interesting option to wrap our approach in a semi-supervised setting, using a small set of labeled links for automatic parameter tuning.

References

1. Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In *International Semantic Web Conference (ISWC)*, 2013.
2. Riccardo Albertoni and Asunción Gómez Pérez. Assessing linkset quality for complementing third-party datasets. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 52–59. ACM, 2013.
3. Mennatallah Amer and Markus Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)*, pages 1–12, 2012.
4. Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15. ACM, 2013.
5. Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *Principles of Data Mining and Knowledge Discovery*, pages 15–27, 2002.
6. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
7. Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. 29(2):93–104, 2000.
8. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), 2009.
9. Gerard de Melo. Not quite the same: Identity constraints for the web of linked data. In *Proceedings of the American Association for Artificial Intelligence*, 2013.
10. Arnab Dutta, Christian Meilicke, and Simone Paolo Ponzetto. A probabilistic approach for integrating heterogeneous knowledge sources. In *Extended Semantic Web Conference*, 2014.
11. Markus Goldstein. Anomaly detection. In *RapidMiner – Data Mining Use Cases and Business Analytics Applications*. 2014.

12. Christophe Guéret, Paul Groth, Claus Stadler, and Jens Lehmann. Assessing linked data mappings using network measures. In *Extended Semantic Web Conference (ESWC)*, pages 87–102. Springer, 2012.
13. Harry Halpin, PatrickJ. Hayes, JamesP. McCusker, DeborahL. McGuinness, and HenryS. Thompson. When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In *The Semantic Web – ISWC 2010*, pages 305–320. Springer Berlin Heidelberg, 2010.
14. Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.
15. Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.
16. Aidan Hogan, Axel Polleres, Jürgen Umbrich, and Antoine Zimmermann. Some entities are more equal than others: statistical methods to consolidate linked data. In *4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS2010)*, 2010.
17. Malte Kiesel and Gunnar Aastrand Grimnes. Dbtropes—a linked data wrapper approach incorporating community feedback. In *EKAW 2010 Demo & Poster Abstracts. International Conference on Knowledge Engineering and Knowledge Management (EKAW-10), 17th International Conference on Knowledge Engineering and Knowledge Management, October 11-15, Lisbon, Portugal*, 2010.
18. Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, and Roland Cornelissen. Test-driven evaluation of linked data quality. 2014.
19. Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1649–1652. ACM, 2009.
20. Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2013.
21. Christian Meilicke. *Alignment Incoherence in Ontology Matching*. University of Mannheim, 2011.
22. Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In *12th International Semantic Web Conference (ISWC)*, 2013.
23. Heiko Paulheim and Johannes Fürnkranz. Unsupervised Generation of Data Mining Features from Linked Open Data. In *International Conference on Web Intelligence, Mining, and Semantics (WIMS'12)*, 2012.
24. Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.
25. Yves Raimond, Christopher Sutton, and Mark B Sandler. Automatic interlinking of music datasets on the semantic web. In *Linked Data on the Web*, 2008.
26. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and Maintaining Links on the Web of Data. In *The Semantic Web - ISWC 2009*, pages 650–665. Springer Berlin Heidelberg, 2009.
27. Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In *Extended Semantic Web Conference*, 2014.
28. Amrapali Zaveri, Dimitris Kontokostas, Mohamed A Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *9th International Conference on Semantic Systems (I-SEMANTICS '13)*, 2013.