DistAD: A Distributed Generic Anomaly Detection Framework over Large KGs

Farshad Bakhshandegan Moghaddam
farshad.moghaddam@uni-bonn.de
University of Bonn
Bonn, Germany

Jens Lehmann
jens.lehmannn@cs.uni-bonn.de
University of Bonn
Bonn, Germany

Hajira Jabeen
hajira.jabeen@gesis.org
GESIS-Leibniz Institute for the Social Sciences
Cologne, Germany

***Abstract*—The last decades have witnessed significant advancements in terms of data generation, management, and maintenance especially in the area of data lakes, and heterogeneous data. This has resulted in vast amounts of data becoming available in a variety of forms and formats including RDF. As RDF data have been created by liberal curation methods (e.g. crowd-sourcing and automatic extraction tools with limited restriction and cross-validation on input data), they are prone to various kinds of errors that can be hidden in different dimensions (i.e. subject, predicate, and object level). Detecting those errors not only improves the KGs quality but also makes it possible to detect anomalous events in the data which can be used for subsequent analysis. In this paper, we present DistAD, a generic, scalable, and distributed framework for anomaly detection on large RDF knowledge graphs. DistAD provides a great granularity for the end-users to select from a vast number of different algorithms, methods, and (hyper-)parameters to detect outliers. The proposed framework is fully open-source, well-documented, and fully integrated into the active community project Semantic Analytics Stack (SANSA). The experiments on real-world use cases disclose that the framework is not only able to handle huge RDF data but also able to successfully detect hidden anomalies/outliers in KGs.**

***Index Terms*—RDF Graph, Anomaly Detection, Big Data, Distributed Computing, Scalable Analytics, SANSA**

I. Introduction

Anomaly Detection (AD) is a branch of data mining dedicated to the discovery of uncommon events in datasets and has several high impact applications in sectors such as security, finance, health care, law enforcement, and much more [1]. The goal of anomaly detection is finding an answer to the critical question, "What is intriguing about a dataset?" It refers to the task of identifying data point(s) and patterns that do not conform to the data's previously specified behavior. Although numerous techniques for detecting outliers and anomalies (anomaly and outlier will be used interchangeably in this paper) in unstructured collections of multiple dimension points have been developed in recent years, yet with the current interest in large-scale heterogeneous data in Knowledge Graphs (KGs), most of the traditional algorithms are no longer directly applicable.

Semantic Web which enables a structural view of the existing data on the web provides machine-readable formats as the Resource Description Framework (RDF) [2]. RDF data are a collection of triples $\langle subject, predicate, object \rangle$ which tend to have rich relationships, forming a potentially very large and complex graph-like structure. Nowadays, many companies in the fields of science, engineering, and business, including energy, bioinformatics, life sciences, business intelligence, and social networks, publish their data in the form of RDF [3][1,2]. Furthermore, the Linked Open Data Project initiative [4] has aided the Semantic Web in gaining greater attention during the last decade. Currently, the Linked Open Data (LOD) cloud has over 10,000 datasets that are available online utilizing the RDF standard[3] and may have sizes up to billions of triples.

KGs are being generated in a variety of ways. Some KGs such as Wikidata [5] and Freebase [6] have been created by crowd-sourcing. NELL [7] has been curated by natural language processing techniques, and DBpedia [8] and YAGO [9] have been created by automatically extracting knowledge tools. Due to the variety of approaches and freedom in inserting the input data, KGs are prone to various kinds of errors because the entered data is neither restricted nor cross-validated. These errors can happen at *Subject*, *Predicate* or *Object* level in the RDF format. For example, there can be extraction errors like parsing errors, e.g. some events in Wikipedia have no starting date, and the value for those events is empty and is written like "- 2001", so extraction tools may interpret this value as a negative year, or there can be errors in the *predicate* level, such as a person has n birth places[4] which $n \gg 1$ (normally, a person has only one birthplace). Moreover, the errors can happen in a multi-feature manner, for example, a person's age can be 5 and it is reasonable, however, the age of a person who is a president of a country can not be 5. So in the multi-feature mode, a combination of different values could yield an anomaly. It is worth mentioning that anomalies are not necessarily wrong values but the values that do not conform with the foreseen data behavior. For example, IoT sensors may generate very high/low values (e.g. temperature), these values are not necessarily wrong but adequate to trigger subsequent actions such as alarms.

Although various strategies for detecting outliers and anomalies have been developed over the years, most standard analytic approaches are no longer directly applicable to KGs due to their graph-like, multi-modal nature, and large

[1] http://www.openphacts.org
[2] https://ontop-vkg.org
[3] http://lodstats.aksw.org/
[4] https://dbpedia.org/page/Alireza_Afzal

size. To tackle the aforementioned issues, in this paper, we propose DistAD, a generic, distributed, and scalable software framework that can automatically detect anomalies in the KGs by extracting semantic features from RDF data, clustering entities, and applying an anomaly detection algorithm on the level of *numeric objects*, *predicates*, and multi-feature scenarios. DistAD offers flexibility over different parts of the workflow and lets the end-users select different approaches and granularity based on their use-cases. In addition, DistAD is integrated into the SANSA Stack [10] and interacts with the different SANSA computational layers. This integration ensures long-term sustainability, as SANSA is an actively maintained project and uses the community ecosystem (i.e. mailing list, issue trackers, continuous integration, and website, etc.).

To summarize, the main contributions of this paper are as follows:

- Introducing a distributed generic framework that can automatically detect anomalies in a large RDF graph
- Integrating the approach into the SANSA Stack
- Covering the code by unit tests, documenting it in Scala docs, and providing a tutorial[5]
- Making DistAD and the framework open-source and publicly available on GitHub[6]
- Evaluation of the results over multiple datasets and empirical evaluation of scalability

The rest of the paper is structured as follows: The related work is discussed in Section II. Preliminaries are presented in Section III. DistAD workflow, and implementation are detailed in Section IV. Section V covers the evaluation of the DistAD and demonstrates the scalability. Finally, we conclude the paper in Section VI.

II. Related work

Although Anomaly Detection is already a well-studied field with the focus specifically on the task of anomaly detection in non-relational datasets [11], however to the best of our knowledge there has not been much dedicated research work particularly on anomaly detection on a large RDF dataset. Therefore in this section, we discuss a few existing outlier detection methods.

One of the pioneers in the area of detecting incorrect numerical data in DBpedia is [12]. The authors performed the process of finding numerical outliers in two steps. In the first step, they grouped the *subjects* based on the *type* information. In next step, the outliers detected by IQR (see Section IV-F). The authors used the FeGeLOD framework [13] for vectorizing the entities and reported that the runtime on datasets containing only two properties- *DBpedia-owl:populationTotal* and *DBpedia-owl:elevation* is over 24 hours due to the slow clustering algorithm.

In another work [14] which is close to [12], the authors introduced two independent outlier detection approaches. In the first phase, they produced a significant sub-population and then applied outlier detection on the sub-population. In the second phase, the *owl:sameAs* property was used to confirm or reject the outliers and detect if an outlier is a natural outlier or not. However, to extract the required information, their method required manual querying of data.

One can consider outlier detection as a quality improvement process over KGs. For example, [15] used a distance-based approach to detect incorrect RDF statements by applying a distance-based clustering method for pointing out the outliers in linked data. The distance was calculated by a semantic similarity measure that took into consideration the *subject type*, *object type*, and the underlying schema.

CONOD [16] is a scalable and generic algorithm for numeric outlier detection for DBpedia. It utilized *rdf:type* and Linked Hypernyms Dataset (LHD) [17] for creating cohorts. Cohorts, unlike clusters, could overlap with each other. For cohorts, [16] used a scalable clustering approach based on Locality Sensitive Hashing (LSH) [18]. As the authors used *rdf:type* and LHD, this approach is only applicable to DBpedia.

In summary, all the above-mentioned methods (except [16]) are not scalable to large-scale knowledge graphs, are complex, and require manual intervention. In addition, although [16] is scalable, but can be only applied on DBpedia. On the contrary, DistAD is distributed and scalable, is automatic and does not require any user interaction, and operates over arbitrary data.

III. Preliminaries

A. Anomaly Detection

The very first and prominent definition of an outlier dates back to 1980, and is given in [19]:

"An outlier is an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism."

As the definition indicates, anomalies are not necessarily wrong values but values that do not conform with the normal data behavior. In the literature, a wide variety of outlier detection approaches have been suggested that can be broadly classified based on nearest neighbors, clusters, or metrics such as density, distance, depth, and statistics. In Section IV-F we discuss a few prominent univariate and statistical methods for outlier detection.

B. Apache Spark

Apache Spark[7] is an open-source unified analytics engine for large-scale data processing. It is available under the Apache Open Source License. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance and provides a set of data analytics modules available in Scala, Python, and Java. Moreover, Spark contains several independent special-purpose libraries such as *Spark Core* which includes basic functionalities of Spark like task scheduling, memory management, fault recovery, and interaction with the storage system, *Spark SQL* which deals with

[5]https://sansa-stack.github.io/SANSA-Stack/

[6]https://github.com/SANSA-Stack/SANSA-Stack/releases/tag/v0.8.3_DistAD

[7]http://spark.apache.org

structured data by providing the DataFrames API, *Cluster Manager* which is designed for distributed computing where parallel operations run on various computer nodes, and last but least MLlib[8] which is a scalable machine learning library to make practical machine learning scalable.

C. SANSA [10]

As we have integrated our framework into the SANSA Stack, in this section, it is briefly introduced. SANSA addresses the need of having a scalable and distributed computational engine to work with semantic data. It benefits from an in-memory analytics framework, Apache Spark, and provides fault-tolerant, highly available, and scalable approaches to efficiently process RDF data with the support for semantic technology standards. SANSA provides various layers of functionality for semantic data representation, querying, inference, and analytics and is available on GitHub[9].

IV. DISTAD AS A RESOURCE

We now present DistAD, a generic framework for Anomaly Detection in KGs. The framework performs anomaly detection by extracting semantic features from entities for calculating similarity, applying clustering on the entities, and running multiple anomaly detection algorithms to detect the outliers on the different levels and granularity. The output of DistAD is a list of anomalous RDF triples. ***a) Availability:*** All the components of the framework are fully integrated into the SANSA ecosystem within the machine learning layer. The framework is fully available for the community as an open-source GitHub repository[10]. ***b) Impact:*** DistAD offers practitioners from the semantic web domain to detect the outliers in their dataset and improve the quality of the existing dataset. Moreover, thanks to its power in handling huge RDF data, it can be used in IoT domains to detect anomalous events in the RDFized sensor data in many areas such as energy domain[11]. ***c) Usability:*** The framework is not only documented on the code level but we also provide a tutorial on its use [12]. We also have numerous samples available as an example class to significantly assist in try-out barrier.

DistAD offers options to select different algorithms and hyperparameters to prepare a customized pipeline for anomaly detection. Table I lists the possibilities and Figure 1 depicts the high-level system architecture overview. In the following, each step of the framework is explained in detail based on Figure 1.

A. Reading Data (Step 1)

The first step of DistAD is reading the RDF data and loading it into memory as a dataframe. The RDF data can reside in normal file systems or HDFS (Hadoop File System). The framework supports *N-Triple* and *Turtle* file formats.

[8]https://spark.apache.org/mllib/

[9]https://github.com/SANSA-Stack

[10]https://github.com/SANSA-Stack/SANSA-Stack/releases/tag/v0.8.3_DistAD

[11]https://platoon-project.eu/

[12]https://sansa-stack.github.io/SANSA-Stack/

B. Similarity Calculation (Step 2)

Most clustering algorithms require a mechanism to calculate the similarity between different data points. In our framework, we used DistSim [25]. The semantic similarity estimations of DistSim operate based on feature sets. These feature sets are derived from the RDF data by the Feature-Extractor Module, which is implemented as a Transformer. DistSim provides multiple modes for the feature extraction module. However, in this framework, we only use the *predicates* (OT mode in [25]) as main features. In short, in this mode, two entities will be similar if they share many common predicates. This helps the clustering algorithm to group similar entities together. Moreover, we implemented two variations, i.e. *Full* and *Partial* similarity. In *Full* similarity we consider all the existing predicates of an entity to calculate the similarity between entities. In *Partial* mode, we only consider predicates that have numeric literals as objects. Although the accuracy of the *Full* mode is higher due to considering all the available predicates, the *Partial* mode benefits from faster operation due to less number of predicates.

C. Clustering Algorithms (Step 3)

Clustering is a key point in most anomaly detection techniques. The reason why clustering is needed in anomaly detection in KGs is that the traditional methods may gather all values of a certain predicate, such as *dbp:weight* and attempt to detect anomalies for this feature. However, in KGs, comparing a feature from different entity types (e.g., the weight of persons against the weight of vehicles) is logically incorrect. Therefore here we mention two clustering algorithms that have been integrated into the DistAD framework.

a) BisectingKmeans [20]: The bisecting K-Means clustering algorithm is a variation on the standard K-Means algorithm. The method begins with a single cluster containing all of the points. Iteratively, it discovers divisible clusters on the bottom level and bisects each one using k-means until there are k total leaf clusters or none are divisible. To promote parallelism, the bisecting steps of clusters on the same level are grouped. If bisecting all divisible clusters on the bottom level results in more than k leaf clusters, the bigger clusters take precedence.

Besides the algorithm, we integrated the Silhouette method with squared Euclidean distance, which is a heuristic approach to determine the optimal number of clusters in a data set. The Silhouette coefficient value presents a measure of how close each point in one cluster is to points in the neighboring clusters. This measure has a range of $[-1, 1]$, and a higher Silhouette Coefficient score relates to a model with better-defined clusters. Our implementation can automatically select the optional k for the clustering.

b) MinHashLSH: Locality Sensitive Hashing (LSH) [18] is an important class of hashing techniques that is commonly used in clustering, approximate nearest neighbor search, and outlier detection with large datasets. LSH hashes data points into buckets using a family of functions ("LSH families"), so that data points that are near to each other are in the same

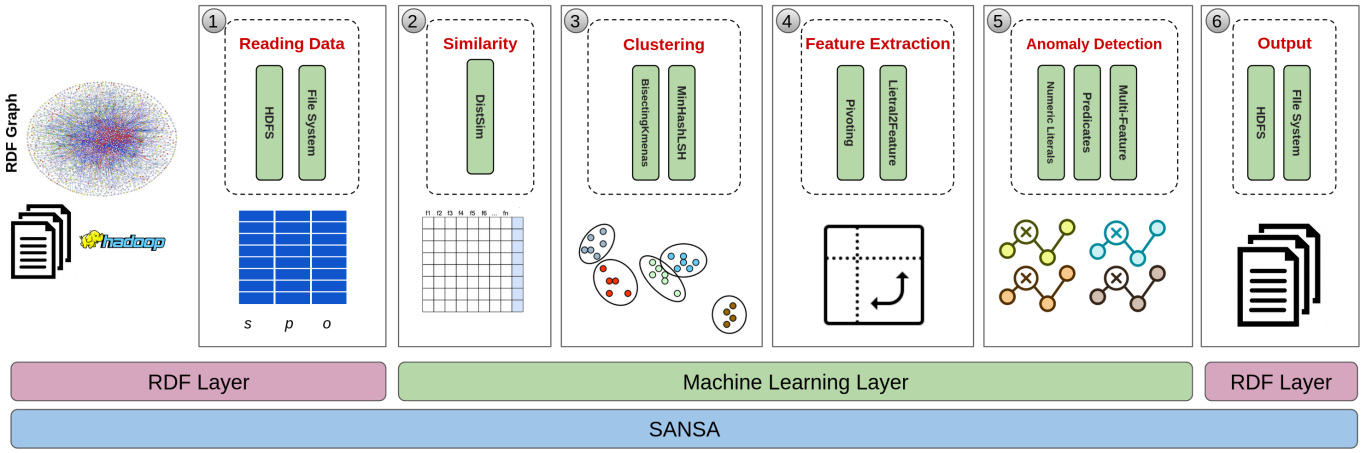

Fig. 1. System Architecture Abstract Overview

TABLE I
DISTAD CONFIGURABLE COMPONENTS

Feature	Options	Comments
Similarity Calculation	`Full Similarity`	Consider all the predicates to generate features
	`Partial Similarity`	Consider only numerical predicates to generate features
Clustering Algorithm	`BisectingKmeans [20]`	Hierarchical version of K-Means clustering algorithm
	`MinHashLSH`	Cohorting based on Local Sensitivity Hash
Feature Extraction	`Pivoting/Grouping`	Basic operation for extracting feature from RDF data
	`Literal2Feature [21]`	Sophisticated method for extracting features from RDF data
Anomaly Detection Type	`Numeric Literals`	Detects anomalies only in the numeric values
	`Predicates`	Detects anomalies on the predicate level
	`Multi-feature`	Detects anomalies on a set of features
Anomaly Detection Algorithms	`Interquartile Range [22]`	Used for single-value features (numeric literals and predicates)
	`Median Absolute Deviation [23]`	Used for single-value features (numeric literals and predicates)
	`Z-score`	Used for single-value features (numeric literals and predicates)
	`IsolationForest [24]`	Used for multi-feature scenarios
Cluster Detection	`Silhouette Method`	For detecting the best optimal number of clusters

bucket with a high likelihood. MinHash[13] is an LSH family method for Jaccard distance where input features are sets of natural numbers. The output of MinHashLSH is a pairwise Jaccard similarity between data points.

D. Feature Extraction (Step 4)

Before being able to apply any anomaly detection algorithm on the RDF data, the KGs should be vectorized (Prepositionalized). This step moves each feature (object, predicate,...) to a separate column in Spark dataframes for further subsequent analysis. To this end, we integrated the following approaches.

a) Pivoting/Grouping: Pivoting is a reshaping mechanism that Spark provides over dataframes. Pivoting reshapes data (produce a "pivot" table) based on column values. The following example depicts how pivoting works on sample RDF data if one wants to pivot the listing 1 based on "Predicate" and aggregate over "Object". The result is shown in listing 2.

```
+-----------------+-------------+-----------+
|Subject          |Predicate    |Object     |
+-----------------+-------------+-----------+
|dbr:Barack_Obama |dbo:birthPlace|dbr:Hawaii |
|dbr:Barack_Obama |dbo:birthDate |1961-08-04 |
|dbr:Angela_Merkel|dbo:birthPlace|dbr:Hamburg|
|dbr:Angela_Merkel|dbo:birthDate |1954-07-17 |
+-----------------+-------------+-----------+
```

Listing 1. Original dataframe before pivoting

```
+-----------------+-------------+--------------+
|Subject          |dbo:birthDate|dbo:birthPlace|
+-----------------+-------------+--------------+
|dbr:Barack_Obama |1961-08-04   |dbr:Hawaii    |
|dbr:Angela_Merkel|1954-07-17   |dbr:Hamburg   |
+-----------------+-------------+--------------+
```

Listing 2. Dataframe after pivoting based on "Predicate" and aggregating over "Object"

b) Literal2Feature: Literal2Feature [21] is a generic, distributed, and a scalable software framework that can automatically transform a given RDF dataset to a standard feature matrix by deep traversing the RDF graph and extracting literals to a given depth. The result of Literal2Feature is a

[13]https://spark.apache.org/docs/latest/ml-features#minhash-for-jaccard-distance

SPARQL query that extracts the features. This option helps the user to extract features that are not in the direct vicinity of an entity for the outlier detection purpose. A possible small sample feature extracting SPARQL query created by the Literal2Feature model is shown in Listing 3. This query is executed by the SANSA built-in SPARQL engine and the result is the vectorized RDF dataframe.

```
SELECT
?movie
?movie__down_title
?movie__down_runtime
WHERE {
?movie a <http://data.linkedmdb.org/movie/film> .
OPTIONAL { ?movie <http://purl.org/dc/terms/title> ?
    movie__down_title .}
OPTIONAL { ?movie <http://data.linkedmdb.org/movie/
    runtime> ?movie__down_runtime .}
}
```

Listing 3. Sample SPARQL query of Literal2Feature

E. Anomaly Detection Type (Step 5)

In DistAD, we have provided 3 types of anomaly detection methods in the framework i.e. a) Numeric Literals b) Predicates c) Multi-Feature

a) Literal Values: Due to the liberty of KG curation methods, KGs are prone to various kinds of errors. As explained earlier, the errors may happen in *Subject*, *Predicate* or *Object* level. For example, there can be extraction errors like parsing errors, e.g. "3-4" can be interpreted as "3", and "-4".

b) Predicates: This type of error happens when an entity has more/less than a usual number of the same predicate. For example, a person normally may have none to a few children. However, if he/she has, for example, 200 children, then this type of (potential) error should be detected to increase the quality of the KGs. In this case, the dataframe containing RDF data is transformed to a new dataframe by grouping based on subjects and predicates and counting based on predicates.

c) Multi-Feature: Multi-feature anomaly detection helps users to detect contextual anomalies. Normally, there exists some hidden correlation between multiple features, for example, there is a positive correlation between the height and the age of a person. By considering these features separately, the algorithm will not detect a person with $1.8m$ height as abnormal. However, having contextual information such as the person's age (e.g., 2 years old) can make this combination anomalous.

F. Anomaly Detection Algorithms (Step 5)

To cover the mentioned anomaly detection methods, we have integrated multiple prominent anomaly detection algorithms. For detecting anomalies in the numerical literals and predicates, IQR, MAD, and Z-Score are implemented, and for the multi-feature scenario, Isolation Forest is integrated.

a) Interquartile Range: The Interquartile Range (IQR) method [22] is a measure of statistical dispersion. It is based on finding the first quartile ($Q1$), the second quartile (Median), and the third quartile ($Q3$) of the given numerical dataset. These quartiles can be clearly seen on a box plot on the data. IQR is the difference between $Q3$ and $Q1$. The data points which are smaller than $Q1 - 1.5 \times IQR$ and greater than $Q3 + 1.5 \times IQR$ are considered as outliers. Figure 2 depicts the IQR and the outliers ranges.

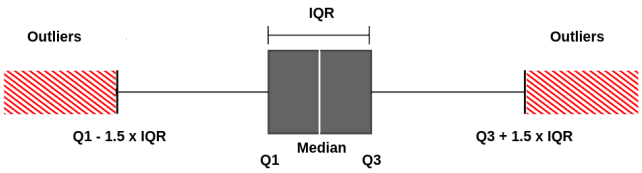

Fig. 2. Box and Whiskers Plot for Interquartile Range

b) Median Absolute Deviation: Median Absolute Deviation (MAD) [23] is a robust measure of the variability of a univariate sample of quantitative data. It is more resilient to outliers in a data set than the standard deviation method. For a univariate data set $X_1, X_2, ..., X_n$ the MAD is defined as the median of the absolute deviations from the data's median, so if $\widetilde{X} = median(X)$ then:

$$MAD = b \times median(|X_i - \widetilde{X}|)$$

where b is a constant scale factor, which depends on the distribution. Any value in the input set X which is greater than $\widetilde{X} + 2.5 \times MAD$ and less than $\widetilde{X} - 2.5 \times MAD$ is considered an outlier.

c) Z-Score: Z-Score is the number of standard deviations by which the value of a raw score (i.e., an observed value or data point) is above or below the mean value of what is being observed or measured. Raw scores above the mean have positive standard scores, while those below the mean have negative standard scores. For example, a Z-score of 2.5 means the data point is 2.5 units away from the mean, so it can be considered as an anomaly. The Z-score is defined as:

$$z - score = \frac{x - mean}{standarddeviation}$$

d) Isolation Forest [24], [26]: Isolation Forest (IF) is an anomaly detection algorithm that identifies anomalies using isolation. Similar to Random Forests, Isolation Forest is built based on decision trees. The algorithm constructs an ensemble of isolation trees from the training data. At the basis of the Isolation Forest algorithm, there is a tendency of anomalous instances in a dataset to be easier to separate from the rest of the sample (isolate), compared to normal points. Randomly sub-sampled data is processed in an Isolation Forest in a tree structure based on randomly selected features. Anomalies are less likely to occur in samples that travel deeper down the tree since they require more cuts to isolate them. Similarly, the samples which end up in shorter branches indicate anomalies as it was easier for the tree to separate them from other observations.

G. Output (Step 6)

The last step of the framework is saving the output to a file. The output is the list of anomalous RDF triples. The triples can be saved as a normal file on a file system or on HDFS.

TABLE II
DATASET STATISTICS

Dataset	Accident	DBpedia1	DBpedia2
Format	Turtle	N-Triple	N-Triple
#Triples	5,961,107	1,000,000	10,000,000
#Distinct Predicates	63	15,520	29,375
#Distinct Numeric Predicates	5	7,067	14,372
File Size	461 MB	137 MB	1.4 GB

H. Implementation

As the programming language of SANSA is Scala[14], we have selected this language and its APIs in Apache Spark to provide the distributed implementation of DistAD. Moreover, we benefit from SANSA IO, ML, and Query layers. Technically, as it can be seen in Figure 1, DistAD can be divided into the following steps 1) Read RDF data as a data frame 2) vectorize the RDF data 3) Cluster the data 4) Extract features for anomaly detection 5) Run anomaly detection algorithm 6) Save the anomalies in a file.

V. EXPERIMENTS

In this section, we present two sets of experiments to analyze different aspects of DistAD. In the first experiment, the correctness of the extracted anomalies will be analyzed over different datasets and in the second experiment, the scalability of the proposed framework will be investigated.

A. Experiment A: Assessment of the detected Anomalies

In this section, we analyze the detected anomalies. To this end, three datasets have been exploited. Engie-accident dataset and 1M and 10M triple sample of DBpedia[15]. Engie SA[16] is a French multinational electric utility company that operates in the fields of energy transition, electricity generation and distribution, natural gas, nuclear, renewable energy, and petroleum. The accident dataset contains data about accidents that occurred in France in 2018[17]. DBpedia dataset is a sample of the infobox part which contains most of the literal predicates in DBpedia. An overview of the datasets is given in Table II.

Without loss of generality, we define the following setups to show the ability of the framework:

a) Anomaly Detection on Numerical Literals: For this case, we use DBpedia1 dataset. We select BisectingKmeans algorithm for clustering, IQR for anomaly detection, *Full* mode for the semantic similarity feature extractor, and use pivoting for vectorization. We also performed the silhouette method and set the number of clusters to 2. As a result, DistAD could detect 19,778 triples that contain anomalous values from 1,232 distinct properties. Some prominent anomalous properties found by DistAD are *dbp:year*, *dbo:postalCode*, *dbo:year2start*, etc. and as it can be seen most of the outliers are found in the date/time related predicates. Manual inspection of detected outliers revealed that the DBpedia extraction tool can not always extract correctly the information related to date/time and that leads to the wrong values. Table III shows a few detected outliers and their corresponding values from Wikipedia.

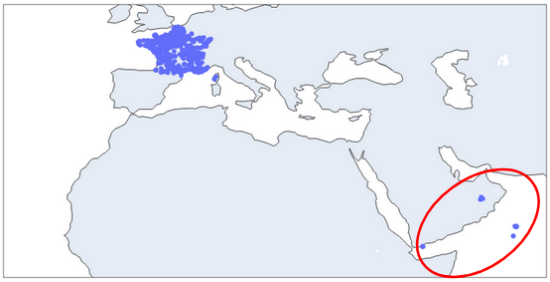

Fig. 3. Accidents which are detected as anomalies

b) Anomaly Detection on Predicates: For this case, we used the same configuration as the previous experiment but use the larger DBpedia2 dataset. By running DistAD, we could detect some interesting anomalies in the predicates. For example, *Nasir_al-Din_al-Tusi* has four *dbo:birthDate* in the DBpedia dump, however, by manually checking it we realized that three out of four are locations, not the dates. This error is propagated to DBpedia because of putting locations under the *Born* property in Wikipedia infobox. Moreover, for example, *Alia_Toukan* has two *dbp:father* which by checking Wikipedia, we realized that there is one name as her father but with two different links which caused this issue.

c) Multi-Feature Anomaly Detection: In this case, we use the Engie dataset. As BisectingKmeans algorithm performs well for clustering, we select it for the clustering algorithm. Regarding anomaly detection, we use IsolationForest and use Literal2Feature [21] (depth=1) for feature extraction. The optimal number of clusters for this dataset is set to 4. Figure 3 shows the detected anomalies. As it can be seen, there are a few accidents that happened outside France and the algorithm could correctly detect them as anomalies by considering their geo-coordinates as a feature set and isolating them from the normal data. These types of anomalies will not be detected if one considers only latitude or longitude separately. However, considering them together make it possible to detect contextual outliers. The dataset contains the data of 55,949 accidents, out of those 1,962 are outside France, and the framework could correctly detect all outliers.

B. Experiment B: Scalability

In this experiment, we evaluate the scalability of DistAD by using different data sizes and varying cluster computing setups. To be able to have different sizes of datasets, we have sampled the DBpedia dataset. Table IV lists the datasets and their characteristic. As the scalability of Literal2Feature and MinHashLSH has been intensively investigated in [21] and [16] respectively, for these experiments, we use BisectingKmeans for clustering, Z-Score for anomaly detection, *Full* mode for the semantic similarity feature extractor, and pivoting/grouping for feature extraction. Moreover, we run

[14]https://www.scala-lang.org/
[15]https://databus.dbpedia.org/dbpedia/collections/latest-core
[16]https://www.engie.com
[17]Can not be publicly published due to Intellectual Property concerns

TABLE III
EXAMPLE OF REAL OUTLIERS IN DBPEDIA

Entity	Predicate	Wikipedia Value	DBpedia Value	Reason
Ian_Turbott	dbp:year	1989-2000	19892000^^xsd:integer	can not handle hyphen
Bidhan_Saran	dbo:postalCode	700004, 700006, 700007	7000047000067000007^^xsd:integer	can not handle commas
Steve_Walters	dbp:year2start	198?–85	198^^xsd:integer	wrong value in Wikipedia

TABLE IV
DATASET DESCRIPTION

Dataset	#Triples	Size
DBpedia-S	1 M	136 MB
DBpedia-M	10 M	1.4 GB
DBpedia-L	50 M	6.8 GB
DBpedia	97.5 M	13.3 GB
DBpedia × 2	195 M	26.6 GB
DBpedia × 4	390 M	53.2 GB

two algorithms for detecting anomalies on numeric literals and predicates.

1) Scalability over number of cores: To adjust the distributed processing power, the number of available cores was regulated. In this experiment, we selected DBpedia (13.3 GB) as a pilot dataset and the number of cores was increased starting from $2^3 = 8$ up to $2^7 = 128$. The experiments were carried out on a small cluster of 4 nodes (1 master, 3 workers): AMD Opteron(TM) CPU Processor 6376 @ 2300MHz (64 Cores), 256 GB RAM. Moreover, the machines were connected via a Gigabit network. All experiments are executed three times and the average value is reported in the results. Figure 4 shows the scalability over different computing cluster setups. It is clear that increasing the computational power horizontally, decreases the execution time. Initially, doubling the number of cores reduces the execution time by nearly a factor of two. However, increasing the number of cores only minimally reduces execution time. This phenomenon is caused by the overhead of moving data between nodes as well as network delay.

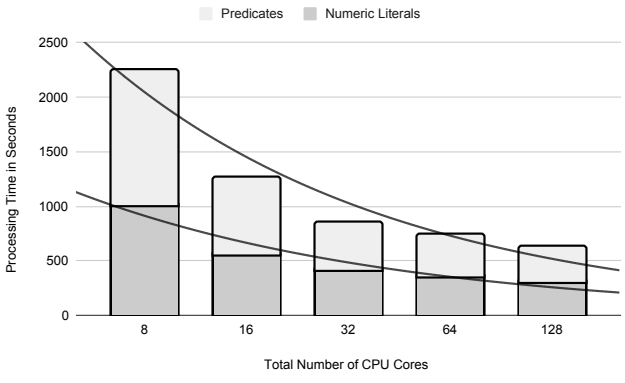

Fig. 4. Processing power vs processing time

2) Scalability over dataset size: To analyze the scalability over different datasets, we fix the computational power to 64 cores and run the experiments for all datasets introduced in Table IV. By comparing the runtime as shown in Figure 5, we note that the execution time does not increase exponentially. Hence, increasing the size of the dataset with the factor of 10 does not necessarily increase the execution time by a factor of 10. This behavior is due to the distribution among available resources e.g. (memory) and partition size. It can be seen that by increasing the dataset size from 1.4 GB to 13.3 GB ($\sim$ 12 times bigger), the execution time almost only doubles.

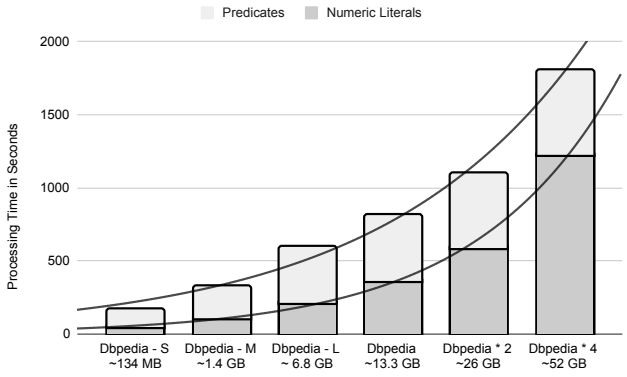

Fig. 5. Sizeup performance evaluation over 64 Cores

VI. CONCLUSION

In this paper, we introduced DistAD, a generic, distributed, and scalable framework for anomaly detection in KGs. DistAD is open-source, available on GitHub, and integrated into SANSA Stack. By providing full control over different algorithms, methods, and (hyper-)parameters, DistAD enables users to have a substantial level of flexibility in using the framework. Our experiments show that the framework can correctly detect different types of anomalies in KGs and it can help to improve the data quality in KGs. Moreover, our experiments show that DistAD can be successfully scaled over a cluster of nodes for a large size of data.

In the future, we plan to work on anomaly explainability over KGs. Especially when performing multi-feature anomaly detection, it is vital to be able to explain why a specific data point is considered an anomaly. Moreover, we plan to work with anomaly detection inductive rules to be able to generate an automatic SPARQL query to fetch all anomalies.

ACKNOWLEDGMENT

This work was partly supported by the EU Horizon 2020 project PLATOON (GA no. 872592). We would also like to thank the SANSA development team for their helpful support.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[2] T. Berners-Lee, "A roadmap to the semantic web," http://www.w3.org/DesignIssues/Semantic.html, 1998.

[3] M. Schmachtenberg, C. Bizer, and H. Paulheim, "Adoption of the linked data best practices in different topical domains," in *The Semantic Web – ISWC 2014*, P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, Eds. Cham: Springer International Publishing, 2014, pp. 245–260.

[4] C. Bizer, M.-E. Vidal, and H. Skaf-Molli, *Linked Open Data*. New York, NY: Springer New York, 2018, pp. 2096–2101.

[5] D. Vrandecic, "Wikidata: a new platform for collaborative data collection," in *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, A. Mille, F. Gandon, J. Misselis, M. Rabinovich, and S. Staab, Eds. ACM, 2012, pp. 1063–1064. [Online]. Available: https://doi.org/10.1145/2187980.2188242

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1247–1250. [Online]. Available: https://doi.org/10.1145/1376616.1376746

[7] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives, "Dbpedia: A nucleus for a web of open data," in *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, ser. Lecture Notes in Computer Science, K. Aberer, K. Choi, N. F. Noy, D. Allemang, K. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, Eds., vol. 4825. Springer, 2007, pp. 722–735. [Online]. Available: https://doi.org/10.1007/978-3-540-76298-0_52

[9] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, Eds. ACM, 2007, pp. 697–706. [Online]. Available: https://doi.org/10.1145/1242572.1242667

[10] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. N. Ngonga, and H. Jabeen, "Distributed semantic analytics using the sansa stack," in *Proceedings of 16th International Semantic Web Conference - Resources Track (ISWC'2017)*. Springer, 2017, pp. 147–155.

[11] C. C. Aggarwal, "Outlier ensembles: Position paper," *SIGKDD Explor. Newsl.*, vol. 14, no. 2, p. 49–58, Apr. 2013. [Online]. Available: https://doi.org/10.1145/2481244.2481252

[12] D. Wienand and H. Paulheim, "Detecting incorrect numerical data in dbpedia," in *The Semantic Web: Trends and Challenges*, V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, and A. Tordai, Eds. Cham: Springer International Publishing, 2014, pp. 504–518.

[13] H. Paulheim and J. Fürnkranz, "Unsupervised generation of data mining features from linked open data," in *2nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, Craiova, Romania, June 6-8, 2012*. ACM, 2012, pp. 31:1–31:12.

[14] D. Fleischhacker, H. Paulheim, V. Bryl, J. Völker, and C. Bizer, "Detecting errors in numerical linked data using cross-checked outlier detection," in *The Semantic Web – ISWC 2014*, P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, Eds. Cham: Springer International Publishing, 2014, pp. 357–372.

[15] J. Debattista, C. Lange, and S. Auer, "A preliminary investigation towards improving linked data quality using distance-based outlier detection," in *Semantic Technology*, Y.-F. Li, W. Hu, J. S. Dong, G. Antoniou, Z. Wang, J. Sun, and Y. Liu, Eds. Cham: Springer International Publishing, 2016, pp. 116–124.

[16] H. Jabeen, R. Dadwal, G. Sejdiu, and J. Lehmann, "Divided we stand out! forging cohorts for numeric outlier detection in large scale knowledge graphs (conod)," in *Knowledge Engineering and Knowledge Management*, C. Faron Zucker, C. Ghidini, A. Napoli, and Y. Toussaint, Eds. Cham: Springer International Publishing, 2018, pp. 534–548.

[17] T. Kliegr, "Linked hypernyms: Enriching dbpedia with targeted hypernym discovery," *Journal of Web Semantics*, vol. 31, pp. 59–69, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570826814001048

[18] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, ser. SCG '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 253–262. [Online]. Available: https://doi.org/10.1145/997817.997857

[19] D. Hawkins, *Identification of Outliers*. Chapman and Hall, 1980.

[20] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *In KDD Workshop on Text Mining*, 2000.

[21] F. B. Moghaddam, C. F. Draschner, J. Lehmann, and H. Jabeen, "Literal2feature: An automatic scalable rdf graph feature extractor," in *Proceedings of the 17th International Conference on Semantic Systems, SEMANTICS 2021, Amsterdam, The Netherlands, September 6-9, 2021*, 2021.

[22] R. McGill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978. [Online]. Available: http://www.jstor.org/stable/2683468

[23] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022103113000668

[24] T. Zemicheal and T. G. Dietterich, "Anomaly detection in the presence of missing values for weather data quality control," in *Proceedings of the 2nd ACM SIGCAS Conference on Computing and Sustainable Societies*, ser. COMPASS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 65–73. [Online]. Available: https://doi.org/10.1145/3314344.3332490

[25] C. F. Draschner, J. Lehmann, and H. Jabeen, "Distsim-scalable distributed in-memory semantic similarity estimation for rdf knowledge graphs," in *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*. Laguna Hills, California: IEEE, 2021, pp. 333–336.

[26] J. Verbus, "isolation-forest," https://github.com/linkedin/isolation-forest, 2021.