

# Measuring Accuracy of Triples in Knowledge Graphs

## Conference or Workshop Item

# Measuring Accuracy of Triples in Knowledge Graphs

Shuangyan Liu, Mathieu d'Aquin, and Enrico Motta

Knowledge Media Institute, The Open University, UK
{shuangyan.liu, mathieu.daquin, enrico.motta}@open.ac.uk

**Abstract.** An increasing amount of large-scale knowledge graphs have been constructed in recent years. Those graphs are often created from text-based extraction, which could be very noisy. So far, cleaning knowledge graphs are often carried out by human experts and thus very inefficient. It is necessary to explore automatic methods for identifying and eliminating erroneous information. In order to achieve this, previous approaches primarily rely on internal information i.e.the knowledge graph itself. In this paper, we introduce an automatic approach, Triples Accuracy Assessment (TAA), for validating RDF triples (source triples) in a knowledge graph by finding consensus of matched triples (among target triples) from other knowledge graphs. TAA uses knowledge graph interlinks to find identical resources and apply different matching methods between the predicates of source triples and target triples. Then based on the matched triples, TAA calculates a confidence score to indicate the correctness of a source triple. In addition, we present an evaluation of our approach using the FactBench dataset for fact validation. Our findings show promising results for distinguishing between correct and wrong triples.

**Keywords:** Data Quality, Triple Matching, Predicate Semantic Similarity, Knowledge Graphs, Algorithm Configuration Optimisation

## 1 Introduction

The concept of Knowledge Graph (KG) was introduced by Google in 2012, to refer to a knowledge base used for enhancing its web-based search results. It is now often used to describe semantic web knowledge bases, i.e. RDF-based representation of some wide domains. Such an RDF representation is known as an RDF triple (subject, predicate, object). An example of RDF triples is (*dbr:Birmingham dbo:populationTotal "1123000"^^xsd:integer*), which represents the fact that the city of Birmingham has a total population of 1123000 (*dbr* and *dbo* are the namespace prefixes of DBpedia repositories).[1] In recent years, several

[1] *dbr* refers to `http://dbpedia.org/resource`, *dbo* points to `http://dbpedia.org/ontology`, *xsd* refers to `http://www.w3.org/2001/XMLSchema#`.

large-scale knowledge graphs have been constructed such as DBpedia[2], YAGO[3], Freebase[4], Wikidata[5], and others.

Many of these knowledge graphs were created by extracting Web contents or through crowdsourcing. These processes could be very noisy, and the created knowledge graphs are unlikely to be fully correct. There is an increasing interest in quality assessment for knowledge graphs [1] [4] [17] [20] [8] [6]. Some approaches focus on completing or correcting entity type information, while others target towards relations between entities, or interlinks between different knowledge graphs. However, research in identifying erroneous literal values automatically is very rare [16]. Proposing a generic approach for measuring the accuracy of triples can identify erroneous information and thus improve the quality of knowledge graphs.

In this paper, we propose the Triples Accuracy Assessment (TAA), an approach for automatically validating RDF triples in a KG (source triples) by collecting consensus of matched triples (among target triples) from other knowledge graphs. A confidence score is assigned to a source triple that is validated to represent the accuracy of this triple. Our approach searches external information for assessing the correctness of triples, which is similar to [8] [6]. The main difference is that we explore other semantic web knowledge bases to find evidence while [6] searches proofs from the Web for validating facts. The main contributions of this paper are presented as follows:

(1) we present an automatic approach that finds consensus from other knowledge graphs for validating the correctness of RDF triples;
(2) we propose a predicate semantic similarity metric based on word-to-word similarity and corpus-based information content;
(3) we enrich our triple validation model to support different types of data including numerical, date and string;
(4) we apply the iterated racing algorithm for tuning the parameters of our system, which finds the best configuration of the parameters of our system.
(5) we evaluate the performance of our approach using gold standard data extracted from a benchmark dataset for fact validation. The findings show that we achieve a competitive F-measure of 95.2% on a train set and 96.1% on a test set.

The initial description of our approach was presented in a workshop paper [12]. The main additions include that we present a new method to compute predicate semantic similarity based on word-to-word similarity and corpus-based information content; we apply the iterated racing algorithm for tuning the parameters of our system, which finds the best configuration of our system; and we also report a systematic evaluation of our approach using a benchmark dataset in this paper.

[2] http://wiki.dbpedia.org/
[3] http://yago-knowledge.org/
[4] https://developers.google.com/freebase/
[5] http://www.wikidata.org

## 2 Related Work

There is a growing body of work on fact validation [6] [20] [10] [5]. This literature can be categorised into two groups in terms of the sources utilised: (1) approaches such as [20] and [10] using internal information (i.e. the knowledge graph itself) for proofs; and (2) approaches such as [6] exploring external information as sources for evidences. Our approach is similar to methods of the second kind, which validate triples using external sources. The main difference from them is that we match evidence triples from other knowledge graphs, not from web documents.

Different methods have been adopted for the fact validation tasks. The approach DeFacto [8] [6] transforms statements into natural language sentences, and retrieves web pages in a web search engine that contain these sentences. A low confidence score is assigned to statements if no or only a few web pages support these sentences. The approaches [20] and [5] apply outlier detection methods to identify errors in numerical property values that are extracted from a data repository. The work [5] improves the prior work by lowering the influence of natural outliers. In more details, [5] performs a second outlier detection on the same property values from equivalent instances to confirm or reject the assessment of a wrong value. However, the work [5] did not address the identification of the same properties of an additional instance for a given instance. Our approach proposes a predicate matching algorithm for finding similar properties of triples.

For predicate matching, we intend to find properties of two different triples which are semantically equivalent but not necessarily syntactically the same. In this paper, we combine a predicate semantic similarity metric and outlier detection techniques for predicate matching, which is different from [3]. The work [3] adopts a string-based similarity method for measuring the similarity between properties.

## 3 The Proposed Approach

TAA is composed of five components (Figure 1). The first two components identify equivalent subject links for a set of source triples, while the middle two components find target triples having matching predicates to the source triples. The last component generates a confidence score for each source triple, representing the level of accuracy of the source triple.

The first component, *Subject Link Fetching* (SLFetching), is used to obtain equivalent links of the subject of a source triple (i.e equivalent subject links). Since non-resolvable and duplicate subject links might be retrieved from the first step, the *Subject Link Filtering* component (SLFiltering) tries to filter out these subject links to achieve an overall efficiency of the subsequent components. Then the *Predicate Object Retrieving* component (POR) collects target triples from external knowledge graphs which contain the identified subject links. In addition, the *Target Triple Matching* component (TTM) combines a set of functions

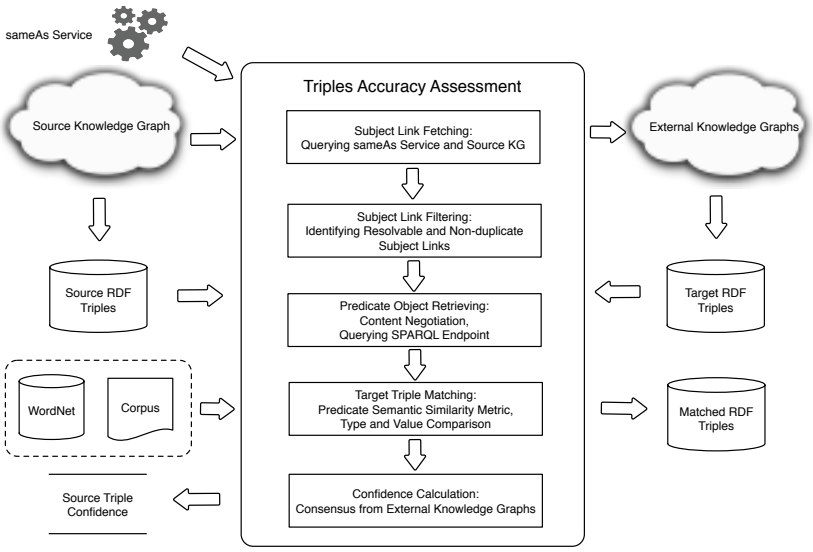


**Fig. 1.** The TAA approach.

for identifying matched triples among the target triples which have predicates semantically similar to the source triple. Finally, the *Confidence Calculation* component (CC) generates the confidence score for the source triple based on agreement among matched triples from different knowledge graphs.

### 3.1 Fetching and Filtering Equivalent Subject Links

The SLFetching component takes the subject of a source triple as input and query the sameAs service[6] and the source knowledge graph to fetch equivalent links for the subject link. The sameAs service can provide equivalent links to arbitrary URIs, and currently serves 200 million URIs. We use the SameAs4J API [7] to fetch equivalent subject links that are provided by the sameAs service. In addition, according to a recent analysis of the LOD cloud datasets [19], *owl:sameAs*[8] is the most commonly used predicate for linking. Hence, we also try to query the source KG using the *owl:sameAs* property. This provides us an alternative way to fetch

[6] sameAs service, `http://sameas.org`
[7] SameAs4J API, `http://99soft.github.io/sameas4j/`
[8] owl is a namespace prefix referring to `http://www.w3.org/2002/07/owl#`.

equivalent subject links when they might not be covered in the sameAs service. Suppose SPARQL query language is implemented in the source KG management system and *subject_uri* denotes the URI link of the subject of the source triple, then the equivalent subject links can be obtained using the following SPARQL query.

```
SELECT ?e WHERE { <subject_uri> owl:sameAs ?e . }
```

Different techniques are combined in the SLFiltering component for filtering the fetched subject links. For identifying a non-resolvable subject link, an HTTP HEAD request is sent to the host where the resource is stored, and the SLFiltering component checks whether a success or redirection HTTP status code can be returned from the host within a given time limit. If not, the subject link is treated as non-resolvable. Meanwhile, an URI equality comparison method is used to clean duplicate subject links.

### 3.2 Retrieving Predicates and Objects

The clean set of subject links come from different sources which serve data in their own formats and provide different access methods. There are different ways of accessing a data repository. Many knowledge repositories e.g. DBpedia, GeoNames[9], LinkedGeoData[10] support content negotiation. Another widely acceptable way is querying SPARQL endpoint. We combine these different data access methods in the POR component for providing a more resilient predicate and object retrieving mechanism. To enable the POR component to handle requests to different SPARQL repositories, we maintain a mapping between common repositories and their SPARQL endpoints. This mapping is provided in a configuration file, which is easy to modify and update.

### 3.3 Matching Target Triples

We combine a predicate semantic similarity metric which is introduced in the following subsection and a predicate type and value comparison algorithm to identify target triples having matching predicates to a source triple. First, we select target triples with predicate similarity that are higher than a given predicate matching threshold ($\alpha$) and remove the target triples which are lower than the given threshold value from the collection of target triples. We then use the selected triples as inputs for the property type comparison procedure, and filter out target triples that have mismatched property types. Then we continue the matching process by applying the predicate value comparison algorithm to remove anomaly triples from the set of target triples. Finally, we obtain the set of matched triples that can be used to examine the accuracy of source triples.

[9] `http://www.geonames.org/`
[10] `http://linkedgeodata.org/`

**Predicate Semantic Similarity** We present a method to compute the semantic similarity between two predicates based on word-to-word similarity and corpus-based information content of words. Information Content (IC) is a measure of concept specificity. More specific concepts (e.g. dog) have higher values of IC than more general concepts (e.g. animal).

We use a matrix to represent the word-to-word similarity of all pairwise combinations of words which constitute the two input predicates. The word-to-word similarity in the matrix is converted from concept-to-concept semantic similarity by taking over the maximal similarity score over all the concepts of the words [18] [22]. Metrics such as [18] [11] [7] [22] can be used to compute the concept-to-concept similarity. In the implementation of TAA, the Wu and Palmer (WUP) method [21] is adopted which measures the depths of the Least Common Subsumer (LCS) of two concepts in the semantic network WordNet [15]. In terms of the concept-to-concept similarity metrics such as the WUP method, WordNet is applied as a concept taxonomy where nodes represent the WordNet concepts or synsets and edges denote hierarchical relations of hypernym and hyponymy between concepts.

We choose the maximal similarity score in a row to represent the similarity of a word in the predicate that the row stands for. Then, we apply the corpus-based information content of the word as the weighting factor to compute the predicate similarity. The definition of corpus-based IC proposed in [18] is presented in Definition 1.

**Definition 1.** *The $IC_{corpus}$ (c) of a concept c is defined as: $IC_{corpus}(c) = -logP(c)$, where P(c) is the probability of encountering the set of instances subsumed by concept c. Let $freq(c) = \sum_{n \in words(c)} count(n)$ be the frequency of concept c occurs in corpus and words(c) is the set of words subsumed by concept c, then $P(c) = \frac{freq(c)}{N}$ where N is the total number of concepts observed in corpus.*

Then the similarity of two predicates is calculated using the predicate similarity metric defined in Eq. (1), which takes the average of the weighted word similarity in either predicate.

$$sim(P_1, P_2) = \frac{1}{2}\left(\frac{\sum_{w \in P_1} maxSim(w, P_2) * IC(c)}{\sum_{w \in P_1} IC(c)} + \frac{\sum_{w \in P_2} maxSim(w, P_1) * IC(c)}{\sum_{w \in P_2} IC(c)}\right), \tag{1}$$

where $maxSim(w, P_2), w \in P_1$ is the maximal word-to-word similarity of a word in the first predicate; $maxSim(w, P_1), w \in P_2$ is the maximal word similarity of a word in the second predicate; and $IC(c)$ is the corpus-based information content of the sense of a word represented by concept c.

In Eq. (1), a word sense disambiguation (WSD) method can be used to find a concept $c$ in a lexicon to represent the sense of a word. In the implementation of

the TAA system, we adopted the Lesk algorithm [9] for word sense disambiguation. The Lesk algorithm determines the sense of a ambiguous word by selecting the concept or synset with the highest number of overlapping words between the context sentence and different definitions from each synset. To determine the concept that represents the sense of a word in a given predicate, the predicate that contains the word is applied as the context information in the Lesk algorithm.

**Predicate Type and Value Comparison** The predicate type and value comparison is intended for filtering out target triples which could be assigned a high predicate similarity score but actually are mismatching. For example, the triples (*dbr:Milton_Keynes dbp:latitude "52.04"*) and (*geodata:2642465 geonames:locationMap http://www.geonames.org/2642465/milton-keynes.html*) have different types of predicates but the two predicates have a high predicate similarity score.[11]

In order to identify the mismatched predicate types, we use a heuristic method. That is, given the predicate type of a source triple, we will first resolve the predicate type that a target triple belongs to and then check whether it matches the format of the source predicate. For numerical type, we check whether the target predicate values conform to a numerical format. In terms of date type, we check whether the target predicate values follow a date pattern (e.g. "yyyy-MM-dd", "yyyy-MM-dd'T'HH:mm:ss'Z'"). For string type, we treat a target predicate as a string type if it does not conform to the numerical type or the date type.

Furthermore, we define a predicate value comparison algorithm (Algorithm 1) to identify mismatched predicates by determining whether there are outliers in the predicate values. We define two procedures for achieving this.

The first procedure which we call *IRange* uses Interquartile Range (IQR) for selecting outliers in a set of predicate values. Many outlier detection methods assume a specific distribution of the datasets to be validated, and the distribution of target predicate values is unknown. Hence, we apply the IQR method here since it is designed for data drawn from a wide range of probability distributions, especially for distributions that are not normal. The concept of the *IRange* procedure is that we first generate the interquartile range, the upper and lower limits of outliers for a set of predicate values, and then determine the outliers in target predicate values if a predicate value is out of the range of $(Q_1 - \varphi * IQR, Q_3 + \varphi * IQR)$. In the implementation of the TAA system, the outlier factor $\varphi$ is set to the value of 1.5 by convention.

The second procedure which we call *SDeviation* is provided to complement the *IRange* procedure for identifying outliers in small sets of data values ($N \leq 4$). The idea of the *SDeviation* procedure is that we first calculate the mean ($m$) and standard deviation ($s$) of a small set of predicate values ($A$), then we consider a data value as an outlier if it is a certain number of standard deviations away

[11] *geodata* and *geonames* refer to `http://sws.geonames.org/` and `http://www.geonames.org/ontology#` repectively.

**Algorithm 1** Predicate Value Comparison

**Precondition:** $A$ is an array containing the property values of a source triple and its target triples

**procedure** IRANGE($A$) ▷ Find outliers in A based on Interquartile Range
  SORT($A$)
  CALCULATEQUARTILES($A$)
  $n \leftarrow A.length - 1$
  **for** $i \leftarrow 0, n$ **do**
    **if** ISOUTLIER($A[i]$) **then**
      MARK($A[i], true$)
    **else**
      MARK($A[i], false$)
    **end if**
  **end for**
**end procedure**
**procedure** SDEVIATION($A$) ▷ Identify outliers for a small A using Standard Deviation
  **if** $A.length > 2$ and $A.length \leq 4$ **then**
    $m \leftarrow$ MEAN($A$)
    $s \leftarrow$ STD($A$) ▷ s is the standard deviation of all property values in A
    $n \leftarrow A.length - 1$
    **for** $i \leftarrow 0, n$ **do**
      **if** $|A[i] - m| \leq \theta \cdot s$ **then** ▷ $\theta$ is the predefined threshold
        MARK($A[i], false$)
      **else**
        MARK($A[i], true$)
      **end if**
    **end for**
  **end if**
**end procedure**

from the mean (noted as $\theta \cdot s$). The standard deviation threshold $\theta$ is set to one in the implementation of the TAA system.

### 3.4 Confidence Calculation

The **C**onfidence **C**alculation component is intended to generate a confidence score based on multiple different matched triples for each source triple to represent its level of accuracy. For triples that have numerical property values, we calculate the confidence score based on a ratio of the difference in property values between a source triple and its matched triples and the weighted average of the matched triples. While for triples that have string property values, we represent the confidence score using a weighted average of the string similarity of the property values between a source triple and its matched triples. We treat date properties as a special case of numerical properties. This is because we can convert a date value into a numerical value representing the number of seconds counted from January 1, 1970, 00:00:00 GMT.

**Table 1.** Rating scale for reliability of subject links

| Rating Score | Definition |
|---|---|
| 1 | very unreliable |
| 2 | unreliable |
| 3 | neutral |
| 4 | reliable |
| 5 | very reliable |

The method to calculate the confidence score for triples having numerical or date type predicate is formulated in Eq. (2):

$$C_{num/date}(x) = 1 - \frac{|x - \gamma|}{|\gamma|} \text{ with } \gamma = \frac{\sum_{i=1}^{m} \omega_i \cdot \nu_i}{\sum_{j=1}^{m} \omega_j} \ , \tag{2}$$

where $x$ is the property value of a source triple; $\gamma$ is the weighted average of the property values of its matched triples; $\omega$ refers to the product of weighting factors including the reliability of the subject link of a matched triple and the predicate similarity between a source triple and a matched triple; $\nu_i$ is the property value of the $i^{th}$ matched triple; and $m$ represents the total number of matched triples obtained.

Furthermore, the method to calculate the confidence score of a triple with a string property value is formulated in Eq. (3):

$$C_{string}(y) = \frac{\sum_{i=1}^{m} \omega_i \cdot y_i}{\sum_{j=1}^{m} \omega_j} \ , \tag{3}$$

where $y_i$ is the string similarity of property values between a source triple and the $i^{th}$ matched triple; $\omega$ refers to the product of weighting factors including the reliability of the subject link of a source triple and the predicate similarity between a source triple and a matched triple; and $m$ is the total number of matched triples obtained.

In Eq. (2) and Eq. (3), the reliability of the subject link of a target triple is rated based on the type of service that is used to fetch the subject link. We define a five-level Likert-scale to represent the reliability of the subject link of a target triple (Table 1). The larger the rating score, the more reliable a subject link is. Two types of services are used in our method: the sameAs service and the source knowledge graph which provides the *owl:sameAs* interlinks. In the implementation of the TAA system, the reliability of the subject link of a target triple which was retrieved from the source knowledge graph was set to be 4 and the equivalent subject links retrieved from the sameAs service was set to be 3. This rating method can also be applied to other services for obtaining equivalent subject links of a source triple.

The confidence score provided by our approach is an indicator for the correctness of triples. A larger value can indicate a higher possibility that a triple is correct. To classify a triple to be correct or wrong, we use a given confidence threshold ($\beta$) to apply to the confidence score.

## 4 Evaluation Methods, Datasets and Experimental Setting

TAA is a parameterised approach since pre-defined values should be provided for the predicate matching threshold ($\alpha$) and confidence threshold ($\beta$) for the system to distinguish between correct and wrong triples. Thus, the performance (evaluated in $F_1$ measure) of TAA can be strongly affected by the specific values taken for the parameters. The goal of the evaluation presented in this paper included: (a) find elite configuration of the parameters that can generate the best performance measured in $F_1$; (b) evaluate the best configuration on a test set which is different from the training set.

We adopted a racing algorithm called iRace [13] to find an appropriate setting of the parameters for TAA. The racing methods for algorithm configuration optimisation were inspired from racing algorithms in machine learning, particularly Hoeffding races [14]. The essential idea of racing algorithms is to evaluate a given set of potential configurations on provided instances, the poor candidate configurations are eliminated as soon as sufficient statistical evidence is gathered, and the race continues only with the surviving ones. The iRace algorithm is an iterative application of F-Race algorithm [2] biasing the sampling of new configurations towards the better candidate solutions at each iteration.

We chose the iterated racing method for our algorithm configuration for three reasons. First, the dependence of TAA's performance on parameter settings is unknown and no explicit model exist to describe the dependence. The iRace method as a model-free algorithm configuration method, can be applied straightaway. Second, the iterated racing methods have been used successfully to automatically configure a variety of state-of-the-art algorithms. Finally, compared with the brute-force approach, the iterated racing approach is more efficient since it does not require repeating the cost evaluation steps for each candidate configuration and poor performing configurations will be discarded as soon as enough statistical evidence is gathered.

To carry out the configuration tuning and validation procedures, we collected gold standard data from the FactBench 2016 benchmark dataset[12]. The collected data are comprised of two subsets: a train set and a test set. The description of the datasets used in experiment is listed below.

- Train set was used in the racing procedure for finding the best configuration for TAA.
  - It consists of 750 triples that were extracted from DBpedia: 150 correct triples and 600 wrong triples.
  - These triples represent two types of relations: date when a person was born and date when a person died. Each triple has either dbo:birthDate or dbo:deathDate as its predicate.
- Test set was applied in the validation procedure for testing the performance of TAA using the best configuration found in the racing procedure.

[12] `https://github.com/SmartDataAnalytics/FactBench`

- It consists of 748 triples that were extracted from DBpedia: 150 correct triples and 598 wrong triples.
- Each triple in the test set has either dbo:birthDate or dbo:deathDate as its predicate.

The racing and validation procedures were carried out using the irace package[13], which implements the iterated racing procedure [13]. The irace package option maxExperiments was set to 1000 as the budget of the experiments for both the tuning and the validation processes. The Friedman test (*F-test*) was used to identify statistically poor performing configurations that can be discarded from the race. The confidence level for the elimination test was set to 0.95. An implementation of TAA was developed and used as the target algorithm to be tuned for irace. An auxiliary program was also implemented for the evaluation, which is called from irace to execute TAA with a specific configuration and instance and return an evaluation value to irace. The evaluation value is the additive inverse of $F_1$ score i.e. $-1 * F_1$. This is because the objective of irace is to minimise the obtained evaluation values, we had to invert the $F_1$ score to maximise the performance of TAA. For predicate matching, we implemented the predicate similarity metric in Eq. (1) based on WordNet version 3.0[14] and NLTK interface.[15] We use the implementation of the word-to-word similarity method in the Sematch framework[16] which adopts the default implementation of the concept-to-concept similarity methods in the WordNet NLTK interface. All the implementations of TAA and resources are available publicly.[17]

## 5 Evaluation Results

For the experiment on the train set, the racing procedure finished after 19 iterations and executed the evaluations for 458 configurations for finding the best configuration. The evaluation results for all the configurations on the train set are plotted in Figure 2.

The results in Figure 2 are grouped by the candidate configuration identifier. For each configuration, the bottom and top of the rectangle represent the minimum and maximum values of evaluation respectively. The middle segment stands for the mean of evaluation values across the instances in the race. Note that the $F_1$ score ranges between 0 and 1, hence, the performance of the TAA system is maximised when the evaluation value equals to $-1$. It is shown that the evaluation values ($-1 * F_1$) have been greatly decreased (dropped below $-0.5$ in general) after the first iteration of the racing procedure which ended at configuration 55. This demonstrates that the performance of TAA have been largely increased after the first iteration.

[13] `http://iridia.ulb.ac.be/irace/`
[14] `https://wordnet.princeton.edu/`
[15] `http://www.nltk.org/`
[16] `https://github.com/gsi-upm/sematch`
[17] `https://github.com/TriplesAccuracyAssessment`

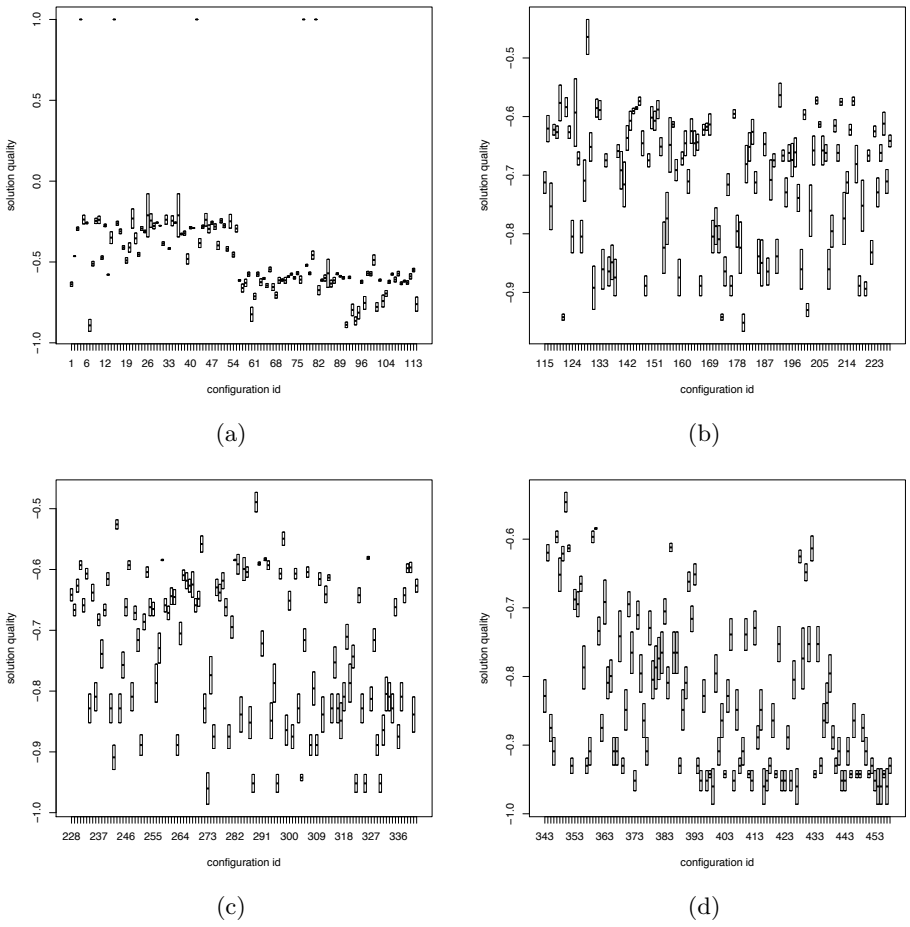


**Fig. 2.** Training results of all configurations in the racing procedure

The best configuration obtained from the racing procedure is the configuration that demonstrates the minimum average evaluation value (i.e. the maximum average $F_1$ score) across different instances. For the racing procedure on the train set, the best configuration *(id=180, $\alpha$=0.812, $\beta$=0.999, mean=*$-0.952$*)* was obtained. For the test set, the best configuration has obtained a mean evaluation value of $-0.961$. The Friedman test results showed that the best configuration is statistically significant different from all the other configurations in the race on both data sets.

During the racing procedure, irace iteratively updated the sampling models of the parameters which enabled the tuning process to focus on the best regions of the parameter search space. The frequency of the sampled configurations is presented in Figure 3. It shows that the configurations with a value between

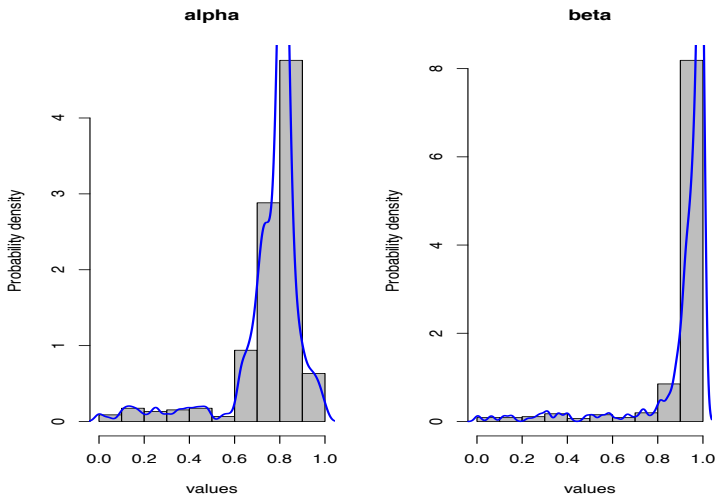


**Fig. 3.** Parameters sampling frequency.

0.8 and 0.9 for the parameter $\alpha$ have the largest density, and are approximately five times as big as the configurations with a value in between 0.6 and 0.7. For the parameter $\beta$, the configurations with a value in between 0.9 and 1.0 have the largest density, and are eight times as big as the configurations with a value between 0.8 and 0.9.

## 6 Conclusions and Future Work

In this paper, we presented an automatic approach, Triples Accuracy Assessment (TAA), for measuring accuracy of RDF triples by checking consensus from different knowledge graphs. We exploit knowledge graph interlinks for discovering equivalent resources, and perform different functions to identify target triples having matching predicates to source triples. This approach supports checking the accuracy of fact triples that have numerical, date or string type properties. The evaluation of the TAA system showed that the best configuration that was identified by the iterated racing procedure is ($\alpha$=0.812, $\beta$=0.999), which demonstrates an F-measure of 95.2% on a train set containing 750 triples from DBpedia and 96.1% on a test set containing 748 triples from DBpedia. The evaluation values obtained for the best configuration is statistically significant different from other candidate configurations on both the train set and the test set.

In the future, we hope to explore how multi-lingual knowledge graph interlinks can be used for fact validation. We also plan to carry out an efficiency evaluation of our approach on large-scale linked data repositories to investigate the scalability of our approach on large-scale linked data.

## References


1. Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Flöck, F., Lehmann, J.: Detecting linked data quality issues via crowdsourcing: A dbpedia study. Semantic Web Journal (to appear)
2. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-race and iterated f-race: An overview. In: Experimental methods for the analysis of optimization algorithms, pp. 311–336. Springer (2010)
3. Cheng, G., Xu, D., Qu, Y.: C3d+ p: A summarization method for interactive entity resolution. Web Semantics: Science, Services and Agents on the World Wide Web 35, 203–213 (2015)
4. Färber, M., Ell, B., Menne, C., Rettinger, A., Bartscherer, F.: Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. Semantic Web Journal (to appear)
5. Fleischhacker, D., Paulheim, H., Bryl, V., Völker, J., Bizer, C.: Detecting errors in numerical linked data using cross-checked outlier detection. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 357–372. Springer, Cham (2014)
6. Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngomo, A.C.N., Speck, R.: Defacto—temporal and multilingual deep fact validation. Web Semantics: Science, Services and Agents on the World Wide Web 35, 85–101 (2015)
7. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. arXiv eprint cmp-lg/9709008 (1997)
8. Lehmann, J., Gerber, D., Morsey, M., Ngomo, A.C.N.: Defacto-deep fact validation. In: The Semantic Web–ISWC 2012, pp. 312–327. Springer (2012)
9. Lesk, M.: Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In: Proceedings of the 5th annual international conference on Systems documentation. pp. 24–26. ACM (1986)
10. Li, H., Li, Y., Xu, F., Zhong, X.: Probabilistic error detecting in numerical linked data. In: Chen, Q., Hameurlain, A., Toumani, F., Wagner, R., Decker, H. (eds.) DEXA 2015, Part I. LNCS, vol. 9261, pp. 61–75. Springer, Cham (2015)
11. Lin, D.: An information-theoretic definition of similarity. In: ICML. vol. 98, pp. 296–304 (1998)
12. Liu, S., d'Aquin, M., Motta, E.: Towards linked data fact validation through measuring consensus. In: 2nd Workshop on Linked Data Quality. vol. 1376 of CEUR Workshop Proceedings (2015)
13. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58 (2016)
14. Maron, O., Moore, A.W.: Hoeffding races: Accelerating model selection search for classification and function approximation. Advances in neural information processing systems 6, 59–66 (1994)
15. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM 38(11), 39–41 (1995)
16. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web Journal 8(3), 489–508 (2017)
17. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. Int. J. on Sem. Web and Info. Sys. (IJSWIS) 10(2), 63–86 (2014)

18. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: 14th Int. Joint Conf. on AI (IJCAI). pp. 448–453. IJCAI/AAAI (1995)
19. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 245–260. Springer, Cham (2014)
20. Wienand, D., Paulheim, H.: Detecting incorrect numerical data in dbpedia. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. pp. 504–518. LNCS, Springer, Cham (2014)
21. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: 32nd annual meeting on Association for Computational Linguistics. pp. 133–138. Association for Computational Linguistics (1994)
22. Zhu, G., Iglesias, C.A.: Computing semantic similarity of concepts in knowledge graphs. IEEE Transactions on Knowledge and Data Engineering 29(1), 72–85 (2017)