Toward Veracity Assessment in RDF Knowledge Bases: An Exploratory Analysis

DIEGO ESTEVES, University of Bonn
ANISA RULA, University of Milano-Bicocca and University of Bonn
ANIKETH JANARDHAN REDDY, Birla Institute of Technology and Science
JENS LEHMANN, University of Bonn and Fraunhofer IAIS

Among different characteristics of knowledge bases, data quality is one of the most relevant to maximize the benefits of the provided information. Knowledge base quality assessment poses a number of big data challenges such as high volume, variety, velocity, and veracity. In this article, we focus on answering questions related to the assessment of the veracity of facts through Deep Fact Validation (DeFacto), a *triple validation* framework designed to assess facts in RDF knowledge bases. Despite current developments in the research area, the underlying framework faces many challenges. This article pinpoints and discusses these issues and conducts a thorough analysis of its pipeline, aiming at reducing the error propagation through its components. Furthermore, we discuss recent developments related to this fact validation as well as describing advantages and drawbacks of state-of-the-art models. As a result of this exploratory analysis, we give insights and directions toward a better architecture to tackle the complex task of fact-checking in knowledge bases.

CCS Concepts: • **Information systems → Retrieval tasks and goals**; *Web searching and information discovery*; • **Computing methodologies → Natural language processing**; *Supervised learning*;

Additional Key Words and Phrases: DeFacto, fact checking, benchmark, exploratory data analysis, trustworthiness, data quality, linked data

ACM Reference format:
Diego Esteves, Anisa Rula, Aniketh Janardhan Reddy, and Jens Lehmann. 2018. Toward Veracity Assessment in RDF Knowledge Bases: An Exploratory Analysis. *J. Data and Information Quality* 9, 3, Article 16 (February 2018), 26 pages.
https://doi.org/10.1145/3177873

1 INTRODUCTION

Creating and managing large-scale knowledge bases (KBs) has been the key to success of many applications. However, if the quality of such KBs is insufficient, this poses a significant obstacle to the uptake of data consumption applications at large scale [44]. Typically, *volume*, *velocity,* and *variety* are used to characterize Big Data. In addition to the aforementioned three characteristics, there exists a fourth characteristic named *veracity* that is gaining importance [3, 37]. The problem of veracity estimation is recognized as one of the key challenges in building and maintaining large

Author's addresses: D. Esteves, University of Bonn, Bonn, 53117, Germany; email: esteves@cs.uni-bonn.de; A. Rula, University of Milano-Bicocca, Italy and University of Bonn, Bonn, 53117, Germany; email: rula@disco.unimib.it; A. J. Reddy, Birla Institute of Technology and Science, India; email: aniketh.reddy@hyderabad.bits-pilani.ac.in; J. Lehmann, University of Bonn, Bonn, 53117, Germany and Fraunhofer IAIS, Bonn, Germany; email: lehmann@cs.uni-bonn.de.

https://doi.org/10.1145/3177873

KBs [38]. This problem is studied in different research communities such as artificial intelligence, databases, and complex systems, and under various names such as *truth-finding*, *fact-checking*, and *trustworthiness* [3].

Fact-checking is a relatively new research area in which algorithms to determine the trustworthiness of an assertion (claim) based on external knowledge sources and common sense rules [32] are designed. These algorithms evaluate text documents, judge whether statements supporting a fact are self-contained in documents, and return a confidence score (or assign a final label) to an input claim. Fact-checking algorithms have diverse real-world applications, ranging from online political debates to news dissemination [18, 43], which are used to estimate the veracity of facts in KBs [22]. In this context, *Trust* and *Provenance* are an essential part of the fact-checking pipeline and they can be explicitly or implicitly represented in the Web of Data [21]. *Provenance* provides a description of the origins of a piece of data and the process by which it has been generated [5]. In other words, it provides meta information about the creation and processing of content [21], making the computation of *trustworthiness* transparent. However, the current Web of Data often omits this meta-information, which is believed to be partially due to the difficulty of locating and evaluating original sources [4]. As a consequence of this shortcoming, the estimation of *trust* is negatively impacted [7], leading to a tedious, time-consuming, and mostly manual fact-validation process that involves finding information by trying out several search queries and evaluating the information present in the websites returned as results to these numerous queries. Though structured, semi-structured, or unstructured data can (technically) be supplied as inputs to these algorithms, we restrict ourselves to fact-checking that takes structured data as input. We refer to that as `Triple Veracity Assessment` task (which encompasses at least one of the following tasks: `Triple Plausibility Estimation`,[1] `Triple Validation`,[2] or `Triple Ranking`[3]). More specifically, we consider the application of fact-checking algorithms to KBs where the knowledge is represented as RDF triples[4] in the form of `(subject, predicate, object)`.

There have been a few approaches proposed to check or validate facts [14, 23, 24, 39, 40]. In this article, we present a systematic review of the components of an existing fact-checking framework named `DeFacto` [14] to understand how errors made by the framework can be mitigated. In addition, we extend the evaluation of `DeFacto` using manually confirmed *facts* obtained from two different KBs, *DBpedia* and *Freebase*. `DeFacto` differs from previous fact-checking approaches as it focuses on calculating the trustworthiness of RDF triples (instead of natural language, i.e., unstructured data) using the web (in contrast to approaches that rely on the RDF graph). Besides assessing the veracity of a given triple, it helps to tackle the problem of missing provenance information by reporting excerpts of web pages that support a given claim.

To the best of our knowledge, such an analysis of a fact-checking framework has not been presented before. Notice that the focus of this work is not to show the performance of `DeFacto`, since this work is already demonstrated in Reference [22], but to extend it over different perspectives that have not been investigated. We aim to find alternatives for the implemented methods that minimize the error propagation. This exploratory analysis will seek to answer the following research questions (RQs) related to `DeFacto`:

— RQ1.1: How does the framework perform on *false* input claims?
— RQ1.2: How sensitive is the accuracy for different *relations* in input facts?
— RQ1.3: Does simple negation add value to the process of *fact-checking*?

[1]The task of verifying whether a triple contains meaningful information or not.
[2]The task of classifying triples in *true* or *false*.
[3]The task of ordering a set of triples by their relevance.
[4]https://www.w3.org/TR/rdf11-concepts.

—RQ1.4: To expand the source of information, can we benefit from Linked Data sources?
—RQ1.5: What is the impact of Named Entity Recognition (NER)?

In addition, we surveyed different `Triple Veracity Assessment` frameworks, comparing their characteristics. This research aims at answering the following research question:

—RQ2.1: How can `Triple Veracity Assessment` models be classified and what are their main features, their advantages, and drawbacks?

Finally, we adapted `DeFacto` to perform the `Triple Ranking` task in addition to `Triple Validation` and evaluated its performance compared to frameworks presented in a recent challenge. Thus, we seek to answer the following question:

—RQ3.1: How well does the current architecture of `DeFacto` perform in `Triple Ranking` tasks?

Therefore, the main contributions of this article are as follows:

(1) We extended the evaluation of one of the state-of-the-art frameworks (`DeFacto`) for the `Triple Validation` task (RQ1.1, RQ1.2, RQ1.3, RQ1.4, RQ1.5).
(2) We presented a survey of `Triple Veracity Assessment` frameworks, i.e., `Triple Validation`, `Triple Ranking`, and `Triple Plausibility Estimation` models (RQ2.1).
(3) We adapted `DeFacto` to perform `Triple Ranking` in addition to `Triple Validation` (RQ3.1).

The rest of this article is structured as follows. In Section 2, we classify, describe, and compare related works. In Section 3, we give an overview of `DeFacto`. In Section 4, we detail the datasets used in this experimental analysis. In Section 5, we provide an exploratory analysis that aims at finding answers to the introduced questions. Finally, we conclude in Section 6, providing guidance, backed by experimental outcomes, for designing `Triple Veracity Assessment` frameworks.

2 RELATED WORK

In general, the task of automated fact-checking can be considered as one of the most challenging tasks in natural language processing (NLP). Apart from designing *trustworthiness* indicators associated with sources of information, the task is especially hard due to the complexity inherent in creating and connecting logical arguments. This is a basis to communicate and defend opinions (or *claims* within this context), to understand new problems and to perform scientific reasoning [36]. Thus, *argumentation mining* methods pose as state-of-the-art solutions for better understanding text structures and relations among entities, i.e., processing raw text in natural language to recover inferential structure [25]. However, it is still a challenging task, and most of the proposed works are of a theoretical nature, lacking useful real-world applications.

With respect to this study, `Triple Veracity Assessment` can be performed in three ways: *validation*, *plausibility*, or *ranking*. Table 1 presents an overview of the features of some state-of-the-art approaches. `DeFacto` is the only *open-source* approach that supports simple counter-evidence searching[5] and also implements metrics to compute the trustworthiness of web sources. The major shortcoming of the system is its dependence on search engines, leading to a higher cost of deployment. However, this disadvantage is common to all *triple validation* architectures. Another major disadvantage is its dependence on predicate expansion methods. Current approaches

[5]We do not rely on complex arguments, but rather in simple evidence that can potentially negate an input claim. For instance, "*James was born in Seattle*" as a counter-evidence to the input claim "`James, born, Paris.`"

Table 1. Triple Assessment Approaches versus Features

Triple Assessment Approaches							
System	Counter Evidences	Real-Time Source	Source Trust.	Reliance on SEs	Nr. Supported Predicates	Predicate Expansion	Open Source
Triple Validation							
DeFacto [14]	Yes	No	Yes	Yes	10	Library	Yes
OpenEval [40]	No	No	No	Yes	Any (upon training)	Keywords	No
KnowItAll [13]	No	No	No	Yes	Many (ontology)	Patterns	No
Triple Ranking							
Bast et al. [2]	No	No	No	No	2	No	No
Bokchoy [10]	No	No	No	No	2	No	Yes
Cress [17]	No	No	No	No	2	No	Yes
Goosefoot [46]	No	No	No	No	2	Synonym-based	Yes
Triple Plausibility							
PAUST [20]	No	No	No	No	Any (ext. resources)	Lexical DBs	No

(1) implement either hard-coded verbalization and rules (fixed or ontology-based), which naturally restricts scalability; (2) use distant supervision methods, which very often have a sub-optimal *precision*; or (3) use external linguistic corpora (e.g., lexical databases) to obtain similar words (e.g., synonyms) to a given *predicate*. It can be observed that these methods are rather crude and hence the verbalizations generated are of a low quality, making verbalization an unsolved task. In the following sections, we describe each task.

2.1 Triple Plausibility Estimation

Plausibility assessment of triples is another related problem. It deals with the measurement of the plausibility of a certain *subject type* being linked to a certain *object type* through a given *predicate*. It could be seen as a prior task to `Triple Veracity Assessment`. Hong et al. [20] propose PAUST, a three-phase system that determines the plausibility of a triple using both DBpedia and Wikipedia as sources of information. Given a set of test triples, PAUST first generates unlabelled training triples by changing the subject, object, predicate, or a combination of the attributes above of the test triples. These changes are made in such a way that subjects, objects, or predicates are replaced with similar subjects, objects, and predicates, respectively. The similarity between various entities is determined using features extracted from DBpedia. WordNet [28], NOMLEX [26], and PreDic [19] are used to determine similarity between predicates. In the second phase, the unlabelled training triples are labeled as plausible or not plausible using the Wikipedia sentence corpus. Using statistical hypothesis testing, PAUST assigns a value between 0 and 1 to a triple denoting the distance between the test triple and the training triple. In the final phase, PAUST determines the plausibility of the test triple by examining the k-nearest neighbors of the test triple. If a majority of the nearest neighbors are plausible, then the test triple is also determined to be plausible. Otherwise it is labeled not plausible. When the subjects, objects, and predicates of many triples belong to the same subject, object, and predicate concepts, respectively, all such triples are given the label that is the majority among such triples.

2.2 Triple Validation

`Triple Validation` is a task in which an input triple is classified as positive or negative, i.e., *true* or *false*. Thus, the process is often performed using supervised classification techniques. There are many approaches and strategies for validating the facts represented by the triples. First, one can search for the input triple on the web, and then apply some method to decide if the triple is true based on the features extracted from the search results [24]. The keywords used while querying the search engines are derived from the *subject* and the *object* of the triple. The web pages retrieved are then ranked based on the calculated values of different features. After determining the feature values and ranks of the search results, the system finally outputs its classification, saying whether the input triple is true or false. These approaches often apply some method to obtain natural language representations (NLRs) of *predicates* (e.g., hard-coded NLRs, string similarity measures, or distant supervision techniques). A different solution is to apply supervised knowledge extraction on the web, and consider a triple as verified if it can be extracted. The approach described in Reference [39] also searches for the triple on the web, where it identifies relevant sources, extracts evidence from them, estimates source trustworthiness, and uses those trustworthiness scores for improving triple evaluation. The difference with the previous works is that it is completely machine learning based. First, a set of data is provided for training the classifiers for each category of triples, instead of using just one classifier for all the categories. This work classifies the set of unlabeled triples to either true or false and provides a confidence value attached to the label. It considers *IS-A* relationships. Furthermore, a third approach may first leverage both search-based and extraction-based techniques to find supporting evidence for each triple and, subsequently, predict the correctness of each triple based on the evidence. The extracted evidence may be from other KBs, and further enriched with evidence from the web and, finally, from query logs. In addition to the other approaches, data fusion techniques are applied for distinguishing correct triples from incorrect ones [23].

One of the earliest systems that leveraged the World Wide Web to validate facts was `KnowItAll` [13]. Soderland et al. [42] describe how `KnowItAll` uses generic patterns and bootstrapping to gauge the confidence of a certain fact. It uses search engine hits to approximate the probability that a certain pattern is correct for detecting a fact that pertains to a given class or relation (a *predicate*). This probability is estimated as the number of web pages returned by a search engine that contain both the pattern and the given fact divided by the total number of pages returned that contain the fact. Each class or relation has multiple patterns and each pattern has an associated probability for a given fact. These probabilities are then fed as features to a naive Bayes classifier, which finally outputs the confidence score of the fact. The main advantage of this method is that the system requires very little supervision because of its bootstrapping capabilities, and can be applied to any generic relation. But a major shortcoming of this approach is its sole reliance on search engines. Many search engines such as Google have now stopped providing APIs that facilitate automatic querying, thereby debilitating this approach. Another major shortcoming of this approach is that it is incapable of measuring the trustworthiness of the source of the information.

`DeFacto` [14] is a system that scores RDF triples based on evidence found in web pages. Though `DeFacto` uses search engines to find evidence, it overcomes the second shortcoming by using a two-pronged approach that takes into account both the trustworthiness of the source and the evidence that supports or contradicts the given fact, thereby improving the quality of its predictions.

`OpenEval` [40] is another fact validation system that leverages Google results to determine the confidence values of a given fact. The system is unique because of its ability to train classifiers within a given time limit (online algorithm). The performance of the classifiers gets better as more time is given for training. `OpenEval` takes as input a set of predicates, a set of seed instances for each of the predicates, and the set of mutually exclusive relationships between the given predicates. For each seed instance, a Google query consisting of the subjects, objects, and the automatically inferred keywords for the predicate is generated. After querying Google, the set of words that occur around the query in the top results is extracted. These sets, called context-based instances (CBIs), are used for training the support vector machines (SVMs) [8]. For each predicate, the set of CBIs generated using the seed instances of that predicate are used as positive examples, and the set of CBIs generated using seed instances of predicates that are mutually exclusive to the given predicate are used as negative examples while training the SVM for that predicate. After training all the SVMs, if there is some time remaining, the SVMs with the maximum entropy are retrained by extracting new CBIs, so as to improve their performance. Another unique aspect is that the keywords used while querying are generated automatically by selecting those words that have the highest weights in the SVMs as the set of possible keywords that represent that predicate. The newly generated keywords are used for generating new CBIs while retraining. While testing, the most important keywords are first used to generate the CBIs. These CBIs are then fed to the appropriate SVM to determine the confidence score of a test instance. If time remains, the keywords with lesser weights are also used while determining the final score. Though this approach has many unique features, it also suffers from its reliance on Google and it is incapable of measuring the trustworthiness of the source of the information. An SVM needs to be trained for each predicate, making it inefficient and time consuming because a set of seed instances and the set of relationships need to be supplied to train each such SVM.

2.3 Triple Ranking (Relevance Scoring)

`Triple Ranking` is the task of ordering triples with the same *subjects* and similar *objects* according to the relevance of the *objects* to the *subjects*. Bast et al. [2] recently explored this problem in detail. Their dataset consisted of manually scored triples whose predicates were either "*profession*" or "*nationality*," and the triples were derived from Freebase. Each triple was scored on a scale from 0 to 7 with 0 indicating least relevance and 7 indicating most relevance. They assumed all triples to be true and built three different triple scoring mechanisms. All of the systems used a related text corpus, which was used to extract features for the classifiers. The first system was based on logistic regression. They trained multiple binary classifiers, one for each profession and one for each nation, which classified triples as primary or secondary. For example, a classifier for the object "Actor" when the predicate is profession would classify the triple having "Tom Hanks" as the subject as primary and would classify the triple having "Barack Obama" as secondary. The second system computed a weighted sum that indicated the degree of relevancy of that object to that subject, predicate pair. This sum was computed by gathering the list of all words that indicated that a given *profession*/*nationality* was the primary *profession*/*nationality* for that person, and then computing a weighted sum of the number of the occurrences of such words with the weights being the TF-IDF values of those words in the related text corpus. The third system used a generative model to assign the probabilities of the triple being relevant based on the related text corpus. The main advantage of the approaches proposed by the authors is that most of the learning happens in an unsupervised manner, which lends the approaches to automation. An important observation is that all of these classifiers require the range of the predicate to be known. Moreover, a classifier needs to be trained for each predicate, object pair. This is not only time and resource intensive but also unfeasible if the range of a predicate is not known or subject to variation. `DeFacto`, on the other hand, does not need to know *a priori* the range of predicates. It uses a single classifier for all triples, leveraging the more generic features mapped by its architecture, thus making it more efficient. However, it is still dependent on a natural language library [15] to obtain the verbalizations for each possible predicate, potentially limiting the approach. Moreover, all of the approaches described in Reference [2] require a related text corpus while training and also for evaluation. This means that the systems cannot handle real-time queries that may need information that is not contained in the related text corpus. `DeFacto` overcomes this major shortcoming by using search engines that provide real-time results, and their results are then used to score triples.

The WSDM Cup 2017 had a challenge that required competitors to build triple ranking models similar to the ones proposed by Bast et al. [2]. Zmiycharov et al. [46] (The Goosefoot Triple Scorer) approached this problem by first downloading related Wikipedia, DeletionPedia, and DBpedia data regarding the persons mentioned in the dataset. They then obtained more training data using distant supervision on the person files. The person files and training data were then normalized using synonym lists for the professions, and nationalities, and other basic transformations. Word2Vec embeddings, TF-IDF features, and Type-like Occurrence Order features were then extracted from the person files for each of the training instances, and a linear regression classifier was then trained using these features. This model was ranked the best according to the Kendall tau metric (*tau*). Though this approach is good for the given task, it is incapable of scoring generic triples because it requires external person files that may not be available.

Hasibi et al. [17] (The Cress Triple Scorer) uses handcrafted features to train a Random Forest model used to predict the relevance score. For each of the relations, these features are extracted from the annotated Wikipedia sentences provided by the challenge. This simple approach performed the best with respect to *average score difference* (AVD) and was ranked second concerning *tau*. Although this approach performs surprisingly well, it does not work for any generic triple since the approach requires handcrafted features for each relation that is not feasible to achieve given the huge number of possible relations.

Bokchoy [10] employed ensemble learning to combine the results of four base scorers—three that used Wikipedia data and one that used Freebase. The three Wikipedia-based classifiers are those proposed in Reference [2]. The main novelty of this scorer was the fourth classifier, which employed Freebase. It was a classifier that predicted the relevance score of a triple based on the path between the subject and the object of the triple in the knowledge base. Positive examples were obtained directly from Freebase and negative samples were generated by randomly replacing real professions/nationalities[6] with other ones and taking care that these replaced professions/nationalities were not associated with the subject. A random forest binary classifier was then trained, which output the score indicating the likelihood of the given predicate connecting the given subject and object. An ensemble was employed to obtain the combined score by computing a weighted sum of the score's output by the base classifiers. The final step involved detecting "trigger" words (manually defined) for a given profession/nationality in the related text for a given person. If trigger words are found in the first paragraph of the Wikipedia text related to that person, the score computed by the ensemble is refined. This approach was ranked the best with respect to *accuracy*, second with respect to *AVD,* and third with respect to *tau index*. Thus, Bokchoy was one of the best classifiers in the challenge. However, it also cannot be applied to any generic triple because it requires a trigger-word-based score tuning, which is not possible for all *relations.* It also suffers from the same shortcomings as those experienced by the systems proposed by Reference [2] since both use the same Wikipedia-based classifiers.

3 DEFACTO IN A NUTSHELL

KBs are built by automatically running information extraction (IE) methods on a variety of sources that may be semi-structured or unstructured. This process of IE is not always error-free. The errors are mainly generated by the extraction process, though errors can creep in, owing to errors in the information provided by the sources themselves. To validate the correctness of facts when building a KB, the system should be able to collect evidence from *complementary sources* where the facts of the KB are mentioned. Originally, DeFacto has been designed such that given an input statement in

[6]The two predicates supported/available in the challenge.

an RDF *triple* format (e.g., `dbr:Albert_Einstein`, `dbo:award`, `dbr:Nobel_Prize_in_Physics`),[7] it finds evidence of the statement on the web. The evidence is a set of web pages, textual excerpts from these pages, and meta-information on the pages that either prove or disprove the fact represented by the input triple. In a nutshell, a summary of `DeFacto` steps is described as follows:

(1) *Pre-processing*: A claim c existing in a knowledge base K (denoted as $c \in K$) is represented by a triple (s, p, o), where s is the subject *uri*, p is the predicate (or *relation*) *uri*, and o is the object *uri*. The function $\gamma(s, p, o, \mathcal{L})$ takes as input a triple (s, p, o) and the set of languages $\mathcal{L}$ and returns a matrix V containing a set of triples. The function γ is calculated as follows:
$\gamma(s, p, o, \mathcal{L}) = [\phi(s, l_1) \times \Gamma(p, l_1) \times \phi(o, l_1)] \cup [\phi(s, l_2) \times \Gamma(p, l_2) \times \phi(o, l_2)], \ldots, \cup[\phi(s, l_n) \times \Gamma(p, l_n) \times \phi(o, l_n)]$, where

(a) $\phi(x, l_i)$ returns a set of m labels $(x_1, x_2, \ldots, x_m)$ that are similar to the label of the resource x ($s \in \mathcal{S}$ and $o \in \mathcal{O}$), which is extracted from the `rdfs:labels` predicate for a given language $l_i \in \mathcal{L}$.
(b) $\Gamma(p, l_i)$ returns a set of verbalized patterns $\mathcal{P}$ for a given predicate p and a language $l_i \in \mathcal{L}$.

The function $\gamma(s, p, o, \mathcal{L})$ returns a matrix V with number of elements $(\mathcal{S} \times \mathcal{P} \times \mathcal{O} \times \mathcal{L})$.

(2)*Information Retrieval*: Afterwards, a set of search engine queries (we call them *metaqueries*) are formalized by concatenating each i^{th} term $(v_i^1, v_i^2, v_i^3) \in V$ without specific search engine parameters (i.e., excepting from the option *market* that defines the location of the retrieved websites and is defined by l, no further parameter is set). The complete retrieval process is carried out by issuing these several queries (the total number of elements of V) to a regular search engine. In the next step, the highest ranked web pages associated with each *metaquery* are retrieved (*evidence sources candidates*).

(3) *Web Page Evaluation*: Once all the web pages have been retrieved, they are processed further, as follows: (a) HTML content is extracted and (b) *fact confirmation* methods are applied to the content extracted (in essence, the algorithm decides whether the web page contains a natural language formulation of the input fact). In addition to *fact confirmation*, the system computes different indicators for the trustworthiness of a web page.[8]

(4) *Final Score (DEFACTO score)*: In addition to finding and displaying sources and their indicators, `DeFacto` also outputs a general discrete confidence value for the input fact that ranges between 0 and 1. `DeFacto` uses features from textual evidence combined with trustworthiness measures to compute the score [14]. It indicates the confidence level of the model for a given input claim. The higher the value, the more likely the input claim is *true*.

A great advantage of `DeFacto` is its ability to handle temporal information. For instance, a triple expressing a relationship may be considered correct just for a certain period of time (e.g., `Tom Cruise`, `marriage`, `Nicole Kidman`, `1987-2001`). Also, compared to other frameworks, `DeFacto` uses the `BOA (BOotstrapping linked datA)` library [15] to generate natural language patterns (verbalizations) from a given predicate (*relation*). Therefore, the use of the library adds more flexibility to the framework. However, this flexibility comes with a reduction in performance since the verbalizations are generated by unsupervised methods. Currently, the library supports 10 *relations*, which does not completely solve the scalability problem. Thus, in general, frameworks for `Triple Veracity Assessment` are dependent on natural language generation tools, which impose barriers due to the limitation in the number of supported *relations*. Furthermore, the lack of more powerful reasoning methods is an important issue. Currently, the *proof extraction* process is

[7]In this example, dbr and dbo stand for the *DBpedia* namespace prefixes http://dbpedia.org/resource and http://dbpedia.org/ontology, respectively.
[8] *Topic Terms*, *Topic Majority in the Web*, *Topic Majority in Search Results,* and *Topic Coverage.*

Table 2. DeFacto Features

	Feature	Description	Advantages	Drawbacks
1	Predicate Expansion	via BOA patterns	Open source	Limited Nr. patterns
2	Proof Extraction	Fixed threshold	Straightforward	Potential to lose valuable information and/or add noise/false positives
3	Temporal Scope	uses REGEX	Straightforward	-
4	Source of Information	The web (Bing)	Largest source of information	Increased financial cost of deployment

Table 3. Gold Standard Datasets: Supported Predicates

Predicate	Description	Temporal	Examples (Test)	Source
FactBench Dataset				
award	persons who received a nobel prize	timepoint	75	Freebase
birth	birth place and date of a person	timepoint	75	DBpedia
death	death place and date of a person	timepoint	75	DBpedia
foundationPlace	place of a company's foundation	timepoint	75	Freebase
leader	presidents of countries	timespan	75	DBpedia
nbateam	team associations of NBA players	timespan	75	DBpedia
publicationDate	author of a book and publication date	timepoint	75	Freebase
spouse	marriage of two persons	timespan	75	Freebase
starring	actors who starred in films	timepoint	75	DBpedia
subsidiary	companies and their subsidiaries	timepoint	75	Freebase
WSDM 2017 Dataset				
profession	profession of a given person	-	513	Freebase
nationality	nationality of a given person	-	197	Freebase

rather based on fixed rules (e.g., string match for objects) instead of argumentative structures [25]. Finally, the dependence on web search results brings a cost of deployment since searching on the web is not free for a high number of query requests. However, this is compensated by the fact that the web is far broader and provides up-to-date content that may not exist in a static corpora. We summarize these DeFacto features in Table 2.

4 DATASETS

We extended the current evaluation [14, 22] of DeFacto through its components (Section 5) using FactBench dataset (Section 5.2) and adapted the framework for the *relevance scoring* task (as discussed in Section 2.3), comparing its performance to other relevant scoring systems (*WSDM2017 challenge*, Section 5.1). In this section, we give a brief overview of these two datasets used in the evaluation. Table 3 describes the predicates used in the datasets in a nutshell.

4.1 WSDM Cup 2017 Triple Scoring Challenge

We adapted and benchmarked `DeFacto` using the dataset provided by the WSDM Cup 2017 Triple Scoring Challenge.[9] The claims in the dataset are in English and the triples were extracted from an April 4, 2014 dump of Freebase. The predicate used in a triple is either *profession* or *nationality*. Each triple is given an integer *score* ranging from 0 to 7 in which 0 and 7 indicate least relevant and most relevant triples, respectively. The task is to rank triples according to their relevance. For example, the triple (*Barack Obama-Profession-Politician*) has a *score* of 7 whereas the triple (*Barack Obama-Profession-Lawyer*) has a *score* of 0. It is worth noting that there are no *false* triples, but just less relevant ones ($scores = 0$). There are 515 *Profession* triples (pertaining to 134 persons) and 162 *Nationality* triples (pertaining to 77 persons) in the training dataset. The test dataset contains 513 *Profession* triples (pertaining to 134 persons) and 197 *Nationality* triples (pertaining to 96 persons). The challenge required contestants to build models that can predict the relevance scores for all the triples. The performance metrics[10] are defined as follows:

— `Average score difference` ($\mathcal{AVD}$): for each triple, take the absolute difference of the relevance score computed by your system and the score from the ground truth; add up these differences and divide by the number of triples.
— `Accuracy` ($\mathcal{ACC}$): the percentage of triples for which the score computed by a given model differs from the score from the ground truth by at most 2.
— `Kendall's tau` ($\mathcal{TAU}$): for each relation, for each subject, compute the ranking of all triples with that subject and relation according to the scores computed by a given model and the score from the ground truth.

The results obtained using `DeFacto` and a discussion about its performance are presented in Section 5.1.

In addition to the training and testing sets, annotated Wikipedia sentences[11] and the Freebase IDs[12] of the entities in the dataset were provided. Also, *gazetteers* for the relations were made available to users to build the models.

4.2 FactBench

FactBench[13] is a gold standard multilingual dataset for fact-checking. It currently provides *claims* in *English*, *German,* and *French*. The claims obtained from two KBs (DBpedia and Freebase) are stored as RDF triples, where each triple represents a single fact and the period (*timespan* or *timepoint*) in which it holds true (for positive examples). The negative examples were derived from the positive examples by modifying them while still following domain and range restrictions. For both positive and negative data (*facts* and *false claims*, respectively), the examples are equally distributed across 10 predicates (p), adding up to 1,500 RDF models[14] (750 for each class, *positive* and *negative*, 75 for each predicate (p)). The period's granularity is a *year*. This means that a *timespan* is an interval between two years, e.g., from 2008 to 2012. A *timepoint* is considered as a *timespan* with the same start and end year, e.g., 2008–2008.

[9] http://www.wsdm-cup-2017.org.
[10] http://www.wsdm-cup-2017.org/triple-scoring.html.
[11] https://dumps.wikimedia.org.
[12] https://www.wikidata.org/wiki/Property:P646.
[13] https://github.com/SmartDataAnalytics/FactBench.
[14] Technically, a model here is a set of triples that contain information about an input triple, e.g., different labels of the same resource for different languages.

5 EVALUATION OF TRIPLE VERACITY ASSESSMENT

In the following sub-sections, we evaluate `DeFacto` against recently proposed `Triple Ranking` approaches (Section 5.1). Further, we extend its analysis [related to (1) pre-processing and (2) information retrieval] through different dimensions that have not been performed yet (Section 5.2), as follows: `proof extraction` (Section 5.2.1), `natural language generation` (Section 5.2.2), `positive and negative example scores` (Section 5.2.3), `negation` (Section 5.2.4), `NER` (Section 5.2.5), and `linked data services` (Section 5.2.6). Configuration files and results are available in a standard format [11] at the project website.[15]

5.1 Triple Veracity Assessment through WSDM

We adapted `DeFacto` to evaluate its performance in the WSDM Cup 2017 Triple Scoring Challenge (Section 4.1). With this aim, we extended `DeFacto` to support the task of `Triple Ranking` in addition to `Triple Validation`. The latter, in its essence, does not achieve a good performance in `Triple Ranking` once it is hard to rank triples when all of them are known *a priori* to be true. To this aim, we included possible *counter evidences* in the pipeline, i.e., seeking for a different entity that can potentially be an *object* candidate in the original triple: $< s, p, o > \Rightarrow < s, p, ? >$, which produces a *set* of triples versus scores $(< s, p, o >_i, score_i)$. Afterwards, we group all triples by $< s >$, order them in a descending order, and apply the formula[16] to each sub-set of grouped triples:

$$f(score, i) = \begin{cases} score & \text{if } i = 0 \\ score * (1 - (i/10)) & \text{otherwise} \end{cases}.$$

In this strategy, we assume that a given person is not very well-known for many of his/her possible *professions/nationalities* and we penalize the final validation *score* for that. Finally, we normalized the final score (see Section 5.2.3) to the $[0 - 7]$ interval as proposed by the challenge (WSDM dataset) and $[0 - 1]$ as proposed by `DeFacto`. It is worth noting that the framework has not been originally designed to rank (`Triple Ranking`), but to validate the veracity of triples (`Triple Validation`). Thus, the triples are classified (*true* or *false*) regardless of any possible correlations among them, which makes *ranking* not optimal. Although (technically speaking) `DeFacto` score can be used directly for the *ranking* task, we observe this independence of triples that does not provide a very good performance, e.g., many (*true*) examples have very close scores (0.90×0.88, i.e., in a range of 100-point percentiles, 2 should not be relevant enough to clearly distinguish the relevance of a set of triples). Thus, in this version we apply a naive function that penalizes the scores grouped by `(subject, predicate)`.

The final score for each triple in the dataset is substituted by this normalized confidence value ($[0 - 7]$). We manually define the verbalization patterns for both relations: *profession* and *nationality*. The benchmark with results is depicted in Table 4.

An expected characteristic of the system is the required processing time: `DeFacto` is much slower compared to most of the frameworks. This is because most of the other frameworks use a static text corpus to extract features for classification while `DeFacto` uses a search engine to gather evidence for and against a given fact on the web. This involves querying the search engine, crawling the websites returned by the search engine, caching the results, and finally extracting evidence and counter-evidence. This web-dependent flow is what causes `DeFacto` to be slower compared to the other frameworks that are run offline. However, once websites were cached, the time required to process the provided dataset reduced to approximately 45 minutes, a more reasonable processing time. Though the higher processing time may seem disadvantageous, usage of the internet

[15] http://smartdataanalytics.github.io/DeFacto.
[16] There are no more than nine triples grouped.

Table 4. Triple Scoring Systems: An Adapted Version of DeFacto Compared to the Three Ranking Systems That Performed the Best at Each Performance Measure in the WSDM 2017 Cup (*Accuracy*, *Average Score Difference*, and *Kendall's tau*)

System (Triple Ranking)	Accuracy	Average Score Difference	Kendall's Tau	Total Runtime
Bokchoy [10]	**0.87**	1.63	0.33	0 : 01 : 15
Lettuce	0.82	1.76	0.36	0 : 02 : 47
Catsear	0.8	1.86	0.41	0 : 05 : 51
Radicchio	0.8	1.69	0.4	0 : 00 : 43
Cress [17]	0.78	**1.61**	0.32	0 : 02 : 30
Samphire	0.78	1.88	0.44	0 : 01 : 05
Chickweed	0.77	1.87	0.39	0 : 00 : 18
Cauliflower	0.75	1.87	0.43	0 : 01 : 32
Goosefoot [46]	0.75	1.78	**0.31**	0 : 01 : 29
Cabbage	0.74	1.74	0.35	0 : 06 : 19
Pigweed	0.74	1.94	0.48	0 : 00 : 02
Fiddlehead	0.73	1.7	0.4	0 : 00 : 02
Rapini	0.73	2.03	0.45	0 : 00 : 20
Chaya	0.7	1.81	0.34	0 : 59 : 09
Gailan	0.7	1.84	0.37	0 : 00 : 07
DeFacto [14]	0.69	1.74	0.39	**4 : 01 : 55**
Celosia	0.69	1.93	0.45	7 : 06 : 31
Kale	0.69	1.85	0.36	0 : 00 : 05
Bologi	0.68	1.91	0.41	0 : 48 : 01
Chicory	0.63	1.97	0.35	0 : 32 : 01
Yarrow	0.6	2.04	0.45	0 : 01 : 11
Endive	0.55	2.49	0.46	0 : 00 : 11

to gather evidence enables DeFacto to validate a wide range of *relations*, in contrast to the very small number of *relations* supported by the compared frameworks (only *nationality* and *profession*). Moreover, competitors are self-limited to their offline corpus during the evidence-searching phase.

Also, given that DeFacto does not use any feature specifically attached to each *relation*, it performs reasonably well compared to its competitors that use handcrafted and custom features. Moreover, DeFacto was originally designed to act as a Triple Validation system and not a Triple Ranking system. On the other hand, the other systems were specifically designed for Triple Ranking and are incapable of performing Triple Validation. In conclusion, DeFacto is more generic and versatile compared to the frameworks developed for the challenge, but this versatility comes at the cost of accuracy and processing time. Finally, it is very important noting that other systems were trained using data dumps of Freebase,[17] which has a high correlation to *Wikipedia* data.

Yet, the benchmark performance shows that there is still room for improvement. In the following section, we explore DeFacto deeper to move toward a better architecture for triple assessment frameworks.

[17] April 14, 2014 dump of Freebase.

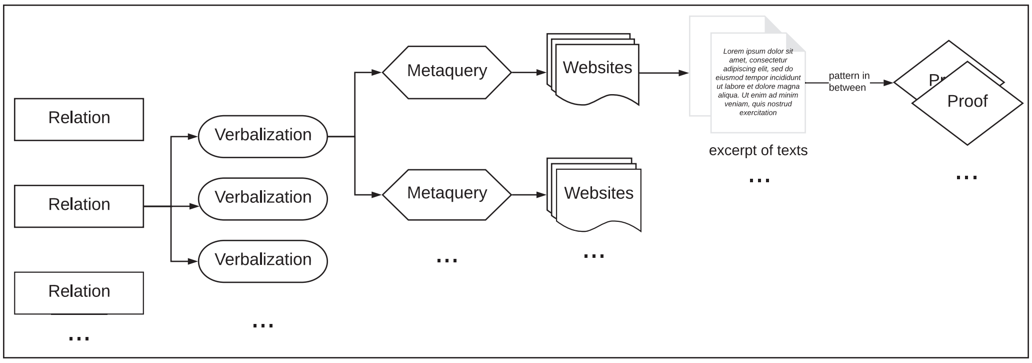
Relation
Relation
Relation
...
Verbalization
Verbalization
Verbalization
...
Metaquery
Metaquery
...
Websites
Websites
...
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
excerpt of texts
...
pattern in between
Proof
...

Fig. 1. The *Proof Extraction* pipeline (*Pattern Verbalization* and *Proof Searching*): extracting excerpt of texts (*proof*) that represent a verbalization for a given *triple*.

5.2 Triple Veracity Assessment through FactBench

In line with the other state-of-the-art approaches discussed in Section 2, DeFacto also imposes restrictions on the number of *relations*, although they are not handcrafted. Yet, more flexibility comes with a performance cost. Moreover, the dependency of search engines implies an increased cost of deployment. The following subsections explore how these issues (among others) impact the current architecture of the framework.

5.2.1 Proof Extraction Pipeline. Before a complete analysis of the underlying steps, we give a formal definition of the current *proof extraction pipeline* implemented as follows:

— *Pattern Verbalization* ($\Gamma(p, l)$): DeFacto uses BOA [15] as an NLP library to verbalize related *predicates* ($v_i^2 \in V$) of a given input claim. Thus, BOA is used as a repository of verbalized patterns $\mathcal{P}_1, \ldots, \mathcal{P}_n$. These patterns allow generating an NLR of the input claim that needs to be verified. The major disadvantage of the pattern verbalization is the limited number of patterns available (e.g, there are verbalization patterns for 10 DBpedia *predicates*), which considerably restricts a *fact-checking* framework. Thus, in case of new claims having different *predicates*, the framework is not able to verify them. Overall, this occurs mainly due to a limitation imposed by the training data. Although a small set of patterns covers most of the simple sentences (e.g., *subject-verb-object* constructions), relevant properties are often spread across clauses or presented in a non-canonical form [1]. This leads most of the pattern extraction approaches based on *supervised* machine learning to fail to cover a reasonable range of pattern sets both in terms of comprehensiveness and quality.

— *Proof Searching*: In a subsequent step, the content of each returned website[18] is extracted using an HTML Parser.[19] The filtering function is applied by searching $\forall v \in V : v^1$ and v^3 within a distance d. If v^2 is found in between (i.e., in the extracted substring), the website w is added to the *proof candidate* set C.

Figure 1 depicts the Proof Extraction pipeline. For a given *relation*, DeFacto obtains the natural language verbalizations (`Verbalization`) from the BOA library, which are then used to generate a set of web search keywords (`Metaquery`). From the retrieved websites (`Websites`) we filter out

[18]In this case, we refer to *websites* as documents.
[19]https://jsoup.org.

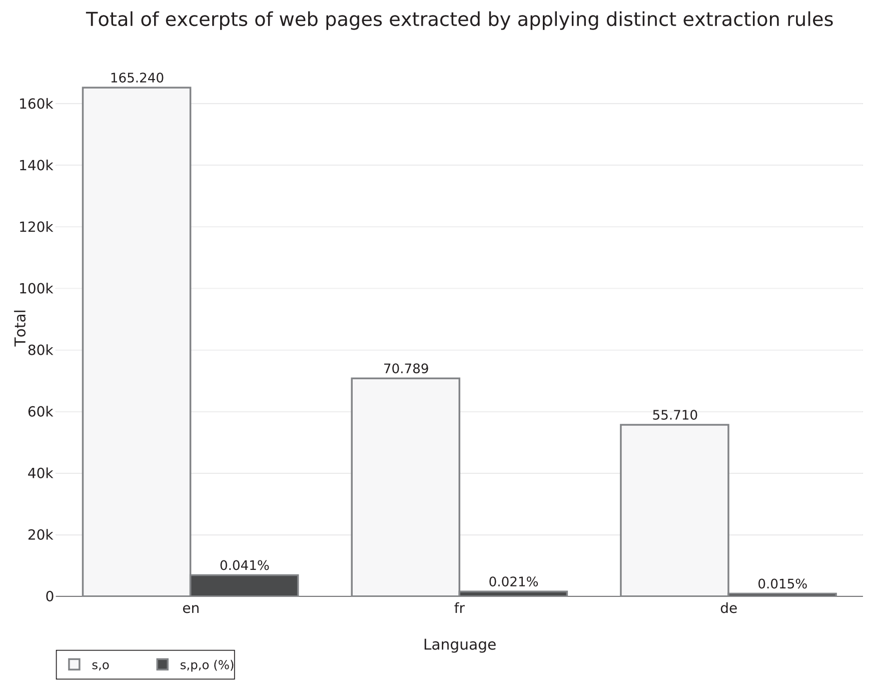

Fig. 2. The exact string match approach: number of *proof candidates* ({*subject, object*}) versus the number of *proofs* ({*subject, predicate, object*}). A drastic reduction in the number of sources.

those that do not have a verbalization of the *relation* (`Proof Candidate`). Finally, we exclude all websites that do not have the verbalization in between *subject* and *object* (`Proof`).

Figure 2 depicts the relationship between the number of *proof candidates* and the final *proofs*, drawing attention to the majority of the content, which is discarded right away, without further analysis.

A qualitative analysis shows that 98.44%, 95.84%, and 97.80% (*German*, *English,* and *French,* respectively) of the retrieved websites are discarded without further content analysis, missing relevant information that can, for instance, argue against a given input claim. For instance, in practical terms, this led to an increase of 6 percentage points in the accuracy of the model for the WSDM dataset evaluation with respect to the original version, which introduces the counter-argumentation analysis (Section 5.1).

5.2.2 Pattern Verbalization: Data Quality Aspects. Despite the simplicity of the triple verbalization process (introduced in Section 5.2.1), the major risk of this strategy refers to losing context and important metadata information. Researchers in linguistics have long pointed out that text is not just a simple sequence of clauses and sentences, but rather follows a highly elaborate structure [29] (in Section 5.2.5 we discuss and present results for this related issue). For instance, from the *relation* dbo:subsidiary we obtain the natural language pattern "*D, the parent company of R*" from the *BOA* library. Although this pattern seems to correctly represent the original *relation*, it may not have a good efficiency at locating excerpts of texts on the web. Table 5 provides statistics about the *Proof Extraction* pipeline shown in Figure 1. The highlighted *predicates* (*gray*) show the low coverage for the verbalization patterns existing in BOA. The number of patterns found between s' and o' holds below μ, where μ is the global mean value of the ratio of good patterns and

Table 5. Triple Verbalization Flow: Number of Websites Returned W, Total of Proof Candidates Pr Extracted from W

Predicate	W	Pr	P_i	$\overline{P_i}$	$P_i/Pr\%$
dbo:award	5448	27244	1897	25,29	6.96
dbo:birth	8357	37776	1010	13,47	2.67
dbo:foundation	2198	7088	110	1,47	1.55
dbo:subsidiary	3013	11730	316	4.21	2.69
dbo:starring	1242	6379	26	0.00	0.41
dbo:death	5970	30839	507	6.76	1.64
dbo:nbateam	7394	42317	1321	17,61	3.12
dbo:publication	9851	35397	1608	21.44	4.54
dbo:leader	7416	38508	1786	23.81	4.64
dbo:spouse	6786	54461	726	9.68	1.33
μ	5768	29174	931	0	2.96

P_i is the Number of Proofs Which Have the Pattern p Verbalized in Between s and o. Absolute Numbers for the Three Supported Languages (*en*, *fr*, and *de*).

the total patterns (P_i/Pr). Considering all *relations*, μ is equal to 2.96. An important finding is that, although the *relations* dbo:foundation or dbo:death have both a low number of high-quality verbalized patterns (one for each), they are more efficient (at locating proofs) than the ones obtained from the *relation* dbo:starring. In this *relation*, although we have at least five good quality patterns, its efficiency is lower than the two aforementioned (0.41 versus 1.55 and 1.64, respectively). These statistics are grouped by predicates. With this analysis, we confront the quality of the patterns versus its real effectiveness to obtain natural language text based on the websites obtained from the web. The efficiency is measured as the number of relevant excerpts of texts that represent a verbalization of an input triple divided by the number of good/high-quality patterns a given *relation* has. Thus, the verbalization set of a *relation* that has one high-quality pattern and returns 100 documents is more efficient than the verbalization set of a *relation* that has 10 high-quality patterns and returns 100 documents. Of course, this efficiency is relative to the number of existing documents and proofs, which is impossible to measure due to the nature of the *Web of Documents*. Yet, they serve as a good indicator of quality for a given *relation*, since we do not consider this efficiency directly related to a relation *per se*. Instead, the efficiency is related to a set of verbalized patterns that are linked to a given *relation*. Therefore, there is a variable that compensates for the rare usage of a given verbalization. Table 6 provides additional basic statistics, confirming the lack of quality and appropriate representations for the emphasized properties. Figure 3 shows details about the manual assessment of the generated natural language verbalizations for each property. The low performance of some properties is due to the high number of incorrect verbalizations.

Whilst a judgment for this event is not simple to establish (and in statistical terms, it is not appropriate to compare the parameters among *relations*), the main conjecture is that the population presents different distributions. Thus the patterns found at the *Web of Documents* (*Web*) were not the same used for training the library (*Web of Data*), making such patterns less efficient than expected. However, this assumption is almost impossible to prove, due to the lack of data availability, the inherent retrieval process of search engines, and the non-deterministic behavior of the web. Another possible cause relies on the difficulty of finding that information on the web. Even though a similar distribution holds, the population for a given *predicate* p is less than μ. Thus, the ratio is

Table 6. Automatically Extracted BOA Patterns per *Property* with Performance Below μ

	dbo:subsidiary	dbo:foundationPlace	dbo:deathPlace
1	R, a subsidiary of D	R based D	D died in R
2	D acquired R	D was born R	D moved to R
3	R merged with D	R law firm D	D was born in R
4	D, the parent company of R	D department store in R	D returned to R
5	R, a division of D	R was renamed D	R author D
6	D's acquisition of R	D between R	D left R
7	R now D	D relocated to R	D joined R
8	R now known as D	D born 1 July 1974 R	R, including D
9	R to form D	D born 26 May 1982 R	R film directed by D
10	D formerly R	D, home R	-
	dbo:starring	dbo:spouse	dbo:birthPlace
1	R film D	R married D	D was born in R
2	R movie D	D, wife R	D returned to R
3	D co-star R	R, his wife D	D left R
4	D actor R	D's marriage to R	R author D
5	D star R	R's husband D	R singer D
6	D, starring R	D, queen consort R	D moved to R
7	D cast member R	D, consort R	R actor D
8	D alongside R	D's widow, R	R writer D
9	D, played by R	R's death, D	D joined R
10	D opposite R	D met R	R artist D

R and *D* stand for *Range* and *Domain*, Respectively (High Quality are Highlighted with *Blue*, Unlikely with *Yellow*, and Incorrect with *White Gray*).

likely to be smaller. Nonetheless, in both theories, due to the content of supported properties, we believe that this is not the major issue (underpinned by *empiric knowledge*). The studied *relations* do not represent specific and restricted knowledge on a certain topic, but are rather generic. Table 6 corroborates to this assumption, evidencing the high number of mistaken verbalization for each *pattern* that presents low performance. As indicated, apart from dbo:spouse, dbo:subsidiary, and dbo:starring, most of the natural language verbalizations (BOA patterns) are not reliable.

Alternatively, collaborative filtering models that learn latent feature representations across surface patterns and structured properties (*universal schemas*) have been proposed by Riedel et. al. [35] and is a candidate for future *fact-checking* enhancement.

5.2.3 `DeFacto` *Score.* For positive examples, `DeFacto` performs well. Figure 4 shows the data distribution for positive examples in FactBench dataset version 2012. For instance, classifying one triple as "true" for any `DeFacto` *final score* greater than 0.8 ($\theta = 0.8$), the model hits more than 90% in positive examples whereas a sharper reduction is observed in the negative examples. Figure 5 shows the *false positive* and *true negative rates*, which are grouped by the *final score*. We report that the major issue is related to the lack of more complex structures to reason with. For instance, the *claim* "*Michael Jackson was born in Houston*" is mistakenly classified as a *fact* (high confidence; *final score* = 0.9) because it was supported by evidence referring to "*Whitney Houston,*" a famous female singer, or even visits to "Houston," the city. The plot draws attention to the high number of *false positives* (166 examples scoring 0.814). Figure 6 depicts the distribution of the scores for

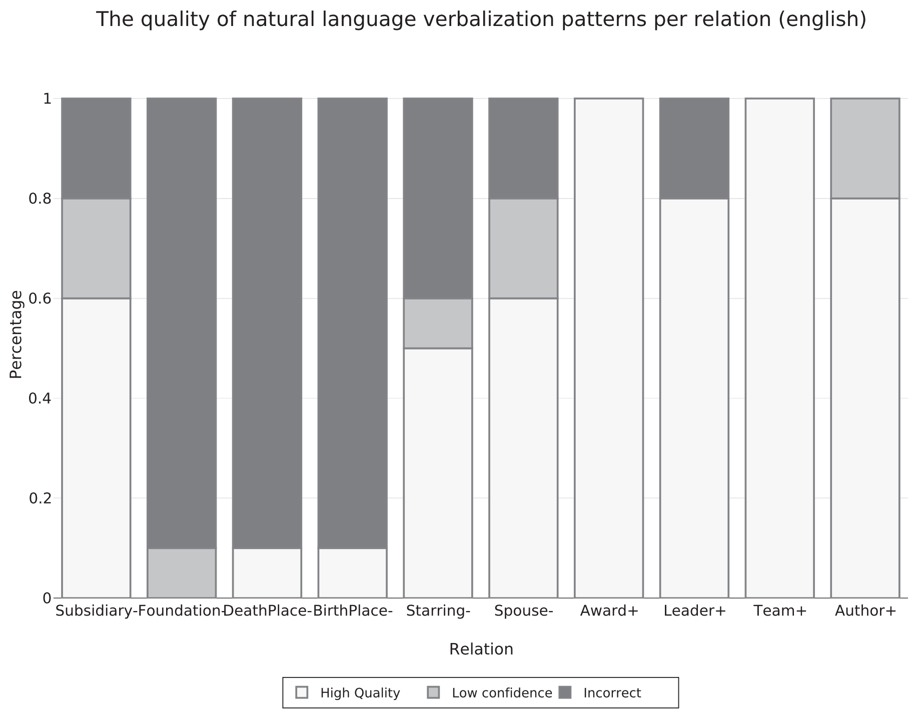

Fig. 3. The quality of natural language verbalization: a manual assessment. The link exposes the relation between patterns with a high number of incorrect derived patterns (*subsidiary, foundation, deathPlace, birthPlace, starring, and spouse*) and values of Pr with ratio between P_i/P_r below μ.

the *negative* sample of FactBench dataset. We observe that the *false positive* rate is relatively high (32.1%) with a cluster of 168 examples being misclassified with score values close to 0.8. Therefore, a small parameter adjustment in the model would considerably increase the final performance of the model. However, the result of any such adjustment would only mask the performance of the model, since such high threshold (e.g., 0.8) would not be adequate to represent negative samples. In the next subsections, we explore further possible improvements.

5.2.4 Simple Negation. Transformations of BOA patterns from FactBench to negative sentences were based upon Part-Of-Speech (POS) tags and WordNet. There are two basic forms of negation—one by injecting a "not" in the sentence and the other by getting an antonym of the BOA English patterns (a more thorough negation case). Positioning of "no" in a sentence is decided by the POS tag placements. The same is the case for generating the antonym patterns, the system needs to figure out which particular word has to be chosen for getting the antonym from WordNet. Negative transformations of sentences with using "not" (or other variant such as "no," "non," "none") is loose negation. Here the negation does not make strong claims. This case of negation indicates that s and o are not related to p. Example: "$\mathcal{D}$ was born in $\mathcal{R}$" $\rightarrow$ "$\mathcal{D}$ was *not* born in $\mathcal{R}$." A further transformation of sentences is made with antonyms and is known as a tight (hard) negation case, i.e., strong claims are made with negation. Now the claim is that s and o are related to the negated (antonym) p. Example: "$\mathcal{D}$ loves $\mathcal{R}$" $\rightarrow$ "$\mathcal{D}$ hates $\mathcal{R}$." It may not necessarily comprehend real natural language formulations. We analyze the impact of adding *opposite verbalization patterns* (*direct* and *antonym*), which may refute input claims by following simple negative assumptions.

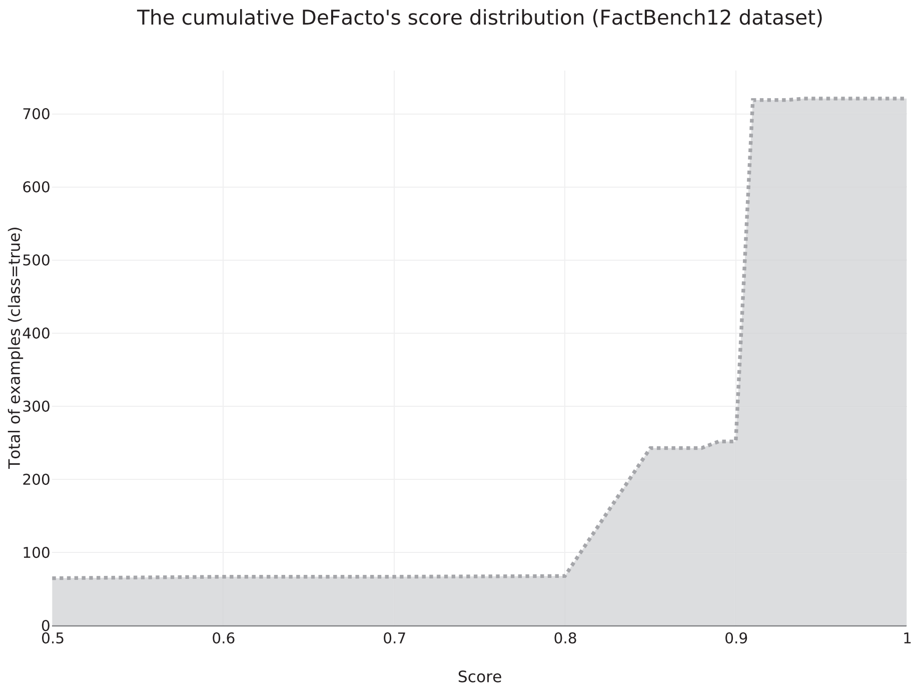

Fig. 4. Score distribution over FactBench (FB) dataset—number of positive examples scored below θ. For $\theta \geq 0.9$ DeFacto has a very high *true positive* rate (total of positive examples in FactBench = 750).

Table 7. Simple Negation: The Number of Negated Patterns (t_p), the Total Websites Retrieved (W), the Total Proof Candidates Pr, and, Finally, the Total Proofs That Have p in between s and o (P_i). Negated Predicate of *Spouse* with Reasonable Absolute Value (198)

Predicate	t_p	W	Pr	P_i	P_i/Pr
dbo:birthPlace	2	1204	2341	2	0.0008
dbo:deathPlace	3	1834	3070	18	0.0058
dbo:foundationPlace	1	179	47	1	0.2128
dbo:spouse	10	1410	5245	**198**	0.0377
dbo:nbateam	3	740	217	5	0.0230

Table 7 shows basic statistics. With the exception of the property dbo:spouse, *direct negation* of *relations* is a strategy that leads to a scenario without further investigation since the *fact-checking* process requires a deeper understanding of the argumentative relations, which is still a challenging research topic (*argumentation mining*). Thus, for many *relations* it is not easy to adopt (i.e., derive a negative structure) while keeping its sense. For instance, the property dbo:born produces diverse valueless negated properties, such as "$\mathcal{D}$ was not born $\mathcal{R}$" (*direct*), "$\mathcal{D}$ was dead $\mathcal{R}$" (*antonym*). The same holds for most of the analyzed properties. In the case of dbo:spouse, some patterns have been shown to be more useful, such as "$\mathcal{D}$ ex-wife $\mathcal{R}$" and "$\mathcal{D}$ divorced $\mathcal{R}$." A further strategy relies on applying logic inference for *functional properties* (Listing 1), i.e., simple rules that seek to search for

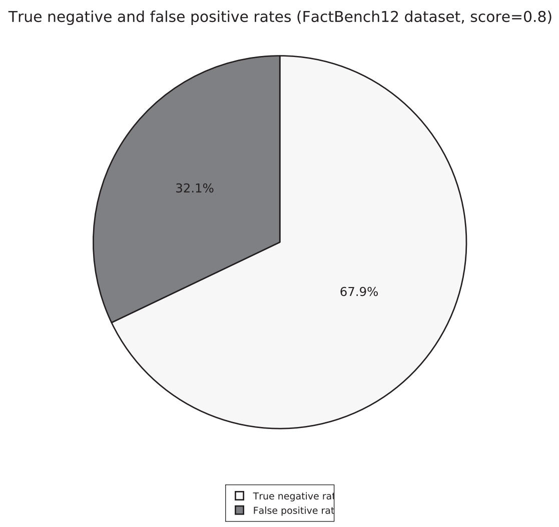

Fig. 5. DeFacto negative examples: *true negative* and *false positive.*

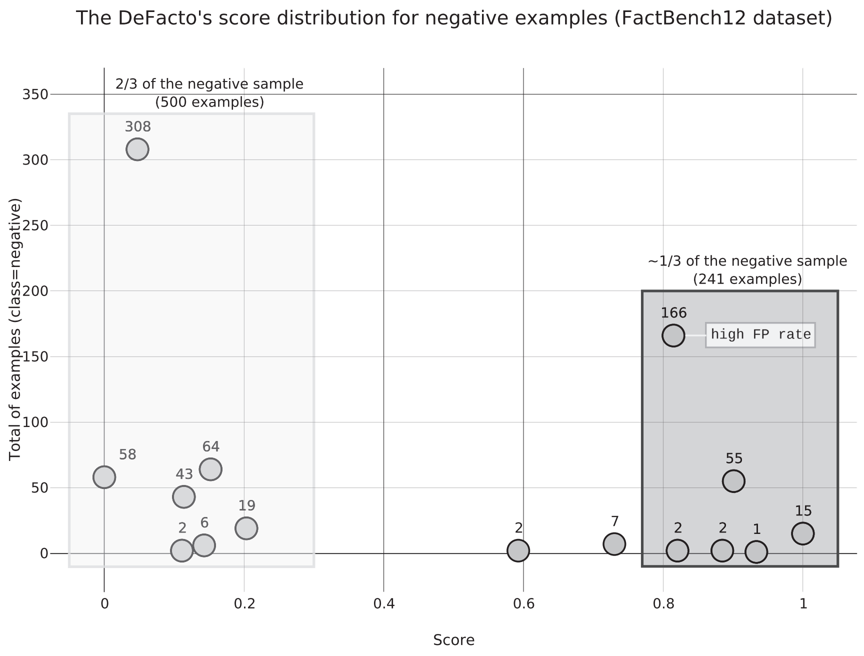

Fig. 6. DeFacto negative examples: scores distribution. Data points under the *light gray* area represent the *true negative* examples whereas under the *dark gray* area represent the *false positive* examples ($\theta \geq 0.8$).

candidates based on omitting $\mathcal{R}$ or $\mathcal{D}$, such as "$\mathcal{D}$ was born $\mathcal{X}$," where $\mathcal{X}$ is any instance of the same type of $\mathcal{R}$ other than o' in a verbalized sentence for a given triple $\{subject \wedge predicate \wedge object\}$.

For instance, "*da Vinci* was born in *Anchiano*" and "*da Vinci* was born in *Tuscany*" would become proof candidates for the false claim, "*da Vinci* was born in *US.*" Thus, alternatives need to be taken into account during the filtering function, obtaining potential sources to argue against the original

```
SELECT DISTINCT ?s ?o2
WHERE {
    ?s rdf:type owl:DatatypeProperty .
    ?s rdf:type owl:FunctionalProperty .
    ?s rdfs:label ?o2 .
    FILTER(langMatches(lang(?o2), "EN"))
} LIMIT 10000
```

Listing 1. SPARQL: Functional Properties defined in DBpedia (English labels).

claim.[20] Finally, a useful *negation* component could be designed with recent *sentiment analysis* techniques [33, 41, 45], where the *polarity* [6] of sentences is computed and taken into account.

5.2.5 Named Entity Recognition (NER). NER is a general task of identifying mentions from text belonging to named-entity types such as *persons* (PER), *organizations* (ORG), and *locations* (LOC). It plays a very important role in the IE pipeline [27, 34] and it is especially challenging in short and informal text contexts, such as emails and text found in the Web of Documents (e.g., *microblogs*) [9, 12]. However, the current version of `DeFacto` neither recognizes nor classifies named entities (NE), but applies *regular expressions* to detect only temporal patterns. The inability to detect NEs and its relations is a downside since `DeFacto` is ignoring relevant information such as structure to judge and better understand texts. Figure 7 depicts the relation between the total number of excerpts of text extracted by applying s and o exact match versus applying the normalized pattern p in between. Results are grouped by named entity class and highlight the strong influence of verbalized patterns. Figure 8 reveals the percentage of final proofs that have at least one NE detected: a drastic reduction to approximately only 4.1% on average of potential NE to be processed. This shows that, at any context size, more than 95% of the proofs are filtered out before applying more complex analysis.[21] According to Pasternack and Roth [30, 31], incorporating *common sense* rules to the fact-checking process enables us to make a more comprehensive and accurate trust decision, e.g., the *relation* dbo:foundationPlace expects a location for the value of a triple *object*. To this end, the concept of *generalized fact-finding*, which refers to a *fact-checking* algorithm, was introduced. The authors reported results using two datasets (*population* and *books*) and showed the benefits of such an approach. Thus, excluding irrelevant entities is potentially relevant to better retrieve *proof candidates* (i.e., a name of an organization would probably not be relevant to a *claim* about a *marriage relation*). To this end, we analyzed the overall impact of the NER algorithm in the *source candidate selection* function; for each property, simple constraints (*common-sense* rules) (e.g., a dbo:marriage *relation* occurs between two persons). Table 8 details the occurrence of each NE class on both *subject* and *object*. The analysis revealed that simple constraints grounded by these *common sense* rules are able to minimize the error propagation in the framework. Figure 9 summarizes the error propagation by *pattern* (overall 15.6% of the proofs violating *common sense* rules). Surprisingly, three of the most accurate *properties* in terms of natural language verbalization [dbo:nbateam (team), dbo:publication (author), and dbo:leader (office) —Table 5] are the ones that present the highest ratio of error propagation (18.99%, 30.72%, 33.67%, respectively). Therefore, the violation of *common-sense* rules is an important issue to be mitigated, once they always directly affect the performance of the model (e.g., the *relation was born* requires a "location" as *object* value).

[20] A drawback is the low number of properties explicitly defined as *functional*.
[21] NEs annotated using Stanford NER version 3.6.0, available at https://nlp.stanford.edu/software/CRF-NER.html.

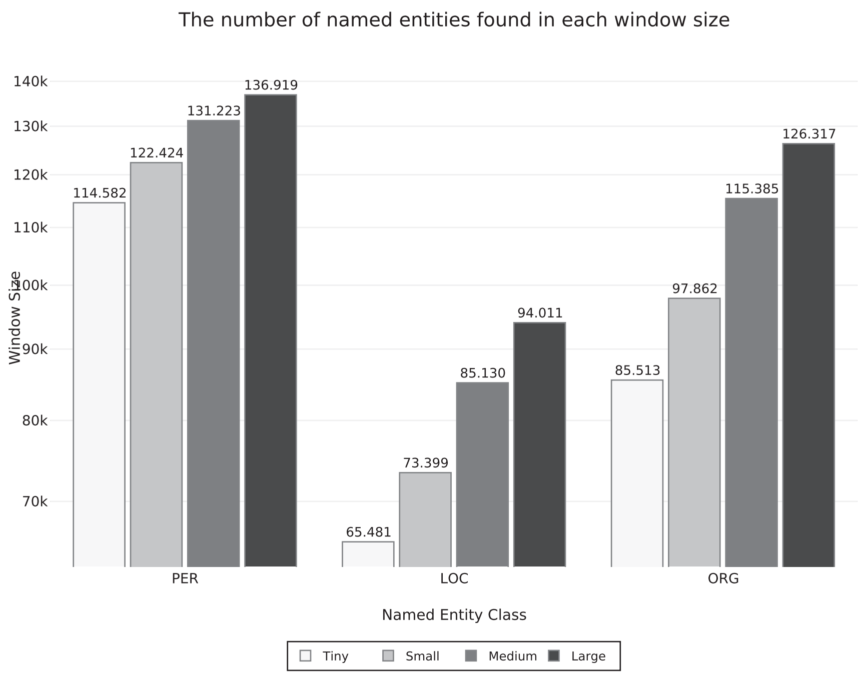

Fig. 7. Proofs and number of named entities (NEs) within different context sizes. Series indicate the context size from 25, 50, 100, and 150 characters to the left and right of the proofs *subject* and *object* occurrence (*Tiny*, *Small*, *Medium,* and *Large*, respectively).

5.2.6 SameAs.org. The Web of Data has many equivalent URIs [16] and, theoretically, these linked URIs could expand the source of information. Thus, this strategy could potentially provide new information. We analyzed the impact of extending the *source selection* function by adding the *SameAs Service*[22] to obtain co-references (`owl:sameAs`[23] property) between different datasets (Definition 5.2.6).

Definition 5.1. Source Candidates. We extend the source candidate search function $S_w = f_{sc}(t, K)$ using the *sameAs* service to include further source candidates, i.e., $\forall r \in R : crawler(t, r)$ where R is the resources returned by the *sameAs* service and $crawler(t, r)$ is a function that extracts text from a given resource r from the given triple t. The final dataset is given by $Sc = S_w \cup S_s$, where S_w is the set of entities obtained from Web and S_s is the set of entities obtained from the *sameAs* service.

This strategy, however, was not beneficial to the pipeline. This is because most of the retrieved *URIs* do not incorporate semantic Web features. This leads to an (insignificant) increase of 0.31% in the number of resources (only 397 out of 126.135 URIs had returned similar linked resources), which declines even more when filtering out *dbpedia* and *wikipedia* domains. Figure 10 shows the most representative *domains* after applying `DeFacto` filters, which shows that the *entertainment* context (e.g., artist and movies websites) is the only one that can slightly add more information to the source candidates.

[22]http://sameas.org.
[23]https://www.w3.org/TR/owl-ref.

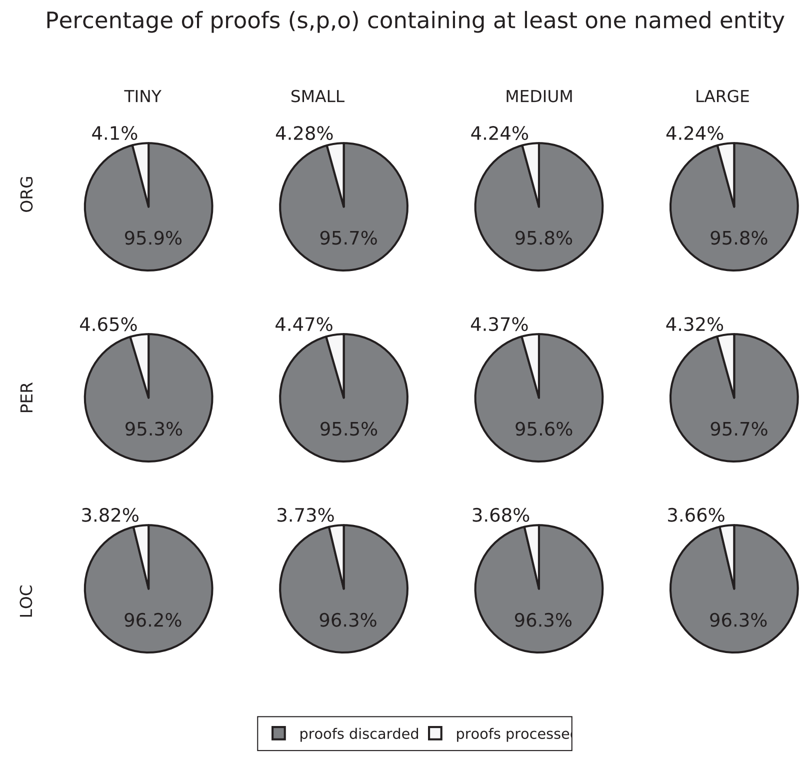

Fig. 8. The ratio between the number of *proof candidates* and *proofs* containing at least one NE (PER, LOC, or ORG) detected: a drastic fall in the number of NE filtered out before revealing more detailed concepts.

Table 8. Classification of *Subject* and *Object* of *Proof Candidates Set*, Including *Common-Sense* Rules

	dbo:award		dbo:birthPlace		dbo:starring		dbo:author		dbo:foundationPlace	
	subject	*object*	*subject*	*object*	*subject*	*object*	*subject*	*object*	*subject*	*object*
PER	3738	5605	3152	5206	947	979	4286	7718	**65**	**47**
LOC	**92**	**183**	6253	4875	**11**	**7**	**2563**	**551**	1485	536
ORG	1519	1996	**389**	**679**	**29**	**51**	1115	1095	640	1837
	dbo:team		dbo:deathPlace		dbo:spouse		dbo:office		dbo:subsidiary	
	subject	*object*	*subject*	*object*	*subject*	*object*	*subject*	*object*	*subject*	*object*
PER	6498	10163	2569	4433	14870	15480	**4134**	**6174**	**250**	**389**
LOC	**2008**	**2644**	5339	3381	**348**	**300**	8597	10766	**155**	**191**
ORG	1506	1676	**263**	**502**	**1281**	**2034**	374	573	4215	5444

Highlighted values are ideal candidates to minimize the *error propagation* based on *common sense* rules. An overview of *English* proofs.

6 CONCLUSION

In this article, we discussed and systematically compared different `Triple Veracity Assessment` frameworks. We focused on general challenges existing for the task, discussing positive and negative aspects of each strategy. In particular, we performed a new experimental analysis of `DeFacto`—a fact-checking framework designed to perform `Triple Validation` in RDF KBs. An important finding is that about half of the patterns do not represent the input's verbalization

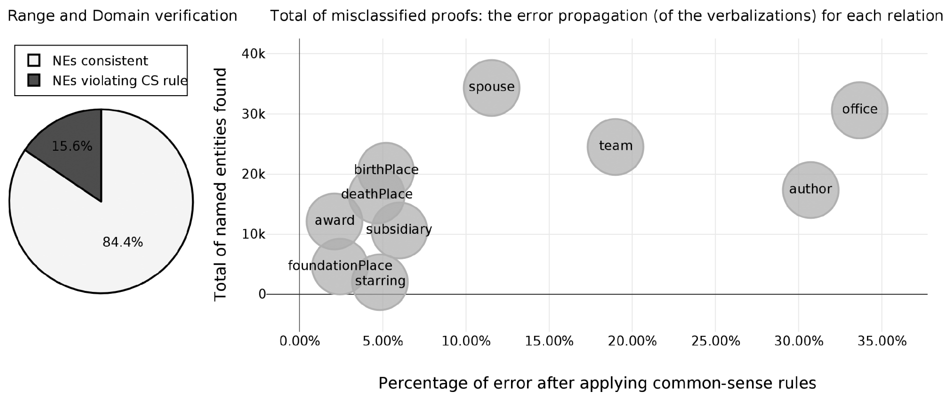

Fig. 9. Common sense rules: an error analysis per *relation*.

Table 9. SameAs.org: Obtaining Information from an External Structured Database

	Filter	Total Resources	%
#1	raw data	122135	1
#2	*sameas.org*	397	0.0031
#3	*.dbpedia.*	122	0.0010
#4	#3 + *.wikipedia.*	92	0.0007

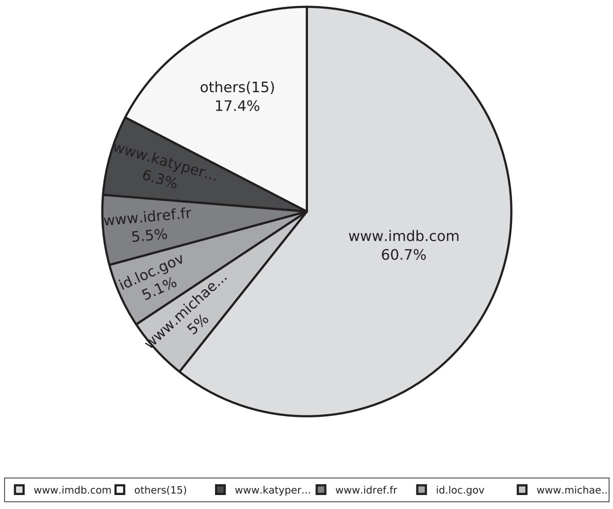

Fig. 10. Sameas.org service: the distribution of most relevant *domains* retrieved using Sameas.org

accurately, leading to an ineffective representation of natural language patterns. This is due to the lack of argumentative structures to reasoning, which leads to poor performance of information retrieval processes. Also, we confirmed that simple negation features do not improve the performance of the framework due to the low level of coverage for the negated patterns. While counterargument functions-based *functional properties* appear to be a suitable strategy to overcome this problem, only 1.69% of the *relations* are labeled as functional (`owl:FunctionalProperty`) in

DBpedia. This can be alleviated by creating a *common sense* layer integrated to the pipeline, including manually generated annotations to validate existing *relations*. Furthermore, extending the set of source candidates using sameas.org creates no value to the process, once most of the retrieved *URIs* do not provide linked data features. Last but not least, we showed that `DeFacto` is able to perform reasonably well in the `Triple Ranking` task, evidencing the good coverage of its model when tested over a different dataset. Overall, we concluded that the major challenge to any `Triple Veracity Assessment` framework relies on the improvement of natural language methods, in particular with respect to the coverage of the verbalization of *relations* (*natural language generation*) as well as the proof extraction phase (*natural language understanding*). To bridge this gap, *argumentation mining* is a promising research area. Finally, in future work, we plan to redesign the `DeFacto` framework to obtain a better estimation of the parameters (e.g., taking the conditional probabilities of related events into consideration) necessary for both `Triple Validation` and `Triple Ranking` tasks.

REFERENCES

[1] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Vol. 1, 344–354.

[2] Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2015. Relevance scores for triples from type-like relations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 243–252.

[3] Laure Berti-Équille and Javier Borge-Holthoefer. 2015. *Veracity of Data: From Truth Discovery Computation Algorithms to Models of Misinformation Dynamics.* Morgan & Claypool Publishers.

[4] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia—A crystallization point for the Web of Data. *J. Web Semant: Sci., Serv. Agents World Wide Web* 7, 3 (2009), 154–165.

[5] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and where: A characterization of data provenance. In *Proceedings of the International Conference on Database Theory.* Springer, 316–330.

[6] Jorge Carrillo de Albornoz, Laura Plaza, and Pablo Gervás. 2010. A hybrid approach to emotional sentence polarity and intensity classification. In *Proceedings of the 14th Conference on Computational Natural Language Learning.* Association for Computational Linguistics, 153–161.

[7] Davide Ceolin, Paul Groth, Willem Robert Van Hage, Archana Nottamkandath, and Wan Fokkink. 2012. Trust evaluation through user reputation and provenance analysis. In *Proceedings of the 8th International Conference on Uncertainty Reasoning for the Semantic Web,* Vol. 900. CEUR-WS.org, 15–26.

[8] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.* 20, 3 (Sept. 1995), 273–297.

[9] Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Inform. Proc. Manag.* 51, 2 (2015), 32–49.

[10] Boyang Ding, Quan Wang, and Bin Wang. 2017. Leveraging text and knowledge bases for triple scoring: An ensemble approach—The BOKCHOY triple scorer at WSDM Cup 2017. In *WSDM Cup 2017 Notebook Papers, February 10, Cambridge, UK*, Martin Potthast, Stefan Heindorf, and Hannah Bast (Eds.). CEUR-WS.org. Retrieved from http://www.wsdm-cup-2017.org/proceedings.html.

[11] Diego Esteves, Diego Moussallem, Ciro Baron Neto, Tommaso Soru, Ricardo Usbeck, Markus Ackermann, and Jens Lehmann. 2015. MEX vocabulary: A lightweight interchange format for machine learning experiments. In *Proceedings of the 11th International Conference on Semantic Systems.* ACM, 169–176.

[12] Diego Esteves, Rafael Peres, Jens Lehmann, and Giulio Napolitano. 2017. Named entity recognition in Twitter using images and text. *Arxiv:1710.11027* (2017).

[13] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in Knowitall: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web (WWW'04).* ACM, New York, 100–110. DOI: http://dx.doi.org/10.1145/988672.988687

[14] Daniel Gerber, Diego Esteves, Jens Lehmann, Lorenz Bühmann, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, and René Speck. 2015. DeFacto—Temporal and multilingual deep fact validation. *J. Web Semant: Sci., Serv. Agents World Wide Web* 35 (2015), 85–101.

[15] Daniel Gerber and Axel-Cyrille Ngonga Ngomo. 2012. Extracting multilingual natural-language patterns for RDF predicates. In *International Conference on Knowledge Engineering and Knowledge Management.* Springer, 87–96.

[16] Hugh Glaser, Afraz Jaffri, and Ian Millard. 2009. Managing co-reference on the semantic web. *WWW2009 Workshop: Linked Data on the Web (LDOW2009).* University of Southampton Institutional Repository. https://eprints.soton.ac.uk/267587/.

[17] Faegheh Hasibi, Darío Garigliotti, Shuo Zhang, and Krisztian Balog. 2017. Supervised ranking of triples for type-like relations—the cress triple scorer at the WSDM Cup 2017. In *WSDM Cup 2017 Notebook Papers, February 10, Cambridge, UK*, Martin Potthast, Stefan Heindorf, and Hannah Bast (Eds.). CEUR-WS.org. Retrieved from http://www.wsdm-cup-2017.org/proceedings.html.

[18] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. 2014. Data in, fact out: Automated monitoring of facts by FactQatcher. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1557–1560.

[19] Soon Gill Hong, Sin-hee Cho, and Mun Yong Yi. 2014. Unsupervised verb inference from nouns crossing root boundary. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers.* Dublin City University and Association for Computational Linguistics, Dublin, 1248–1259.

[20] Soon Gill Hong and Mun Yong Yi. 2017. Plausibility assessment of triples with instance-based learning distantly supervised by background knowledge. Submitted to *Semant. Web J.* Retrieved from http://www.semantic-web-journal.net/system/files/swj1546.pdf.

[21] Krzysztof Janowicz. 2009. *Trust and Provenance—-You Canfit Have One Without the Other.* Technical Report. Institute for Geoinformatics, University of Muenster, Germany.

[22] Jens Lehmann, Daniel Gerber, Mohamed Morsey, and Axel-Cyrille Ngonga Ngomo. 2012. DeFacto—Deep fact validation. In *Proceedings of the International Semantic Web Conference.*

[23] Furong Li, Xin Luna Dong, Anno Langen, and Yang Li. 2017. Knowledge verification for longtail verticals. *PVLDB* 10, 11 (2017), 1370–1381.

[24] Xian Li, Weiyi Meng, and Clement Yu. 2011. T-verifier: Verifying truthfulness of fact statements. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE'11).* IEEE Computer Society, Washington, DC, 63–74. DOI: http://dx.doi.org/10.1109/ICDE.2011.5767859

[25] Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol. (TOIT)* 16, 2 (2016), 10.

[26] Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. NOMLEX: A lexicon of nominalizations. In *Proceedings of Euralex98.* 187–193.

[27] Mónica Marrero, Julián Urbano, Sonia Sánchez-Cuadrado, Jorge Morato, and Juan Miguel Gómez-Berbís. 2013. Named entity recognition: Fallacies, challenges and opportunities. *Computer Standards & Interfaces* 35, 5 (2013), 482–489.

[28] George A. Miller. 1995. WordNet: A lexical database for english. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. DOI: http://dx.doi.org/10.1145/219717.219748

[29] Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law.* ACM, 98–107.

[30] Jeff Pasternack and Dan Roth. 2011. Generalized fact-finding. In *Proceedings of the 20th International Conference Companion on World Wide Web.* ACM, 99–100.

[31] Jeff Pasternack and Dan Roth. 2011. Making better informed trust decisions with generalized fact-finding. In *IJCAI.* 2324–2329.

[32] Jeff Pasternack and Dan Roth. 2013. Latent credibility analysis. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13).* 1009–1020.

[33] Zhong Qian, Peifeng Li, Qiaoming Zhu, Guodong Zhou, Zhunchen Luo, and Wei Luo. Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.* 815–825.

[34] Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning.* Association for Computational Linguistics, 147–155.

[35] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics, 74–84.

[36] Gil Rocha, Henrique Lopes Cardoso, and Jorge Teixeira. 2016. ArgMine: A framework for argumentation mining. *Computational Processing of the Portuguese Language-12th International Conference, PROPOR.* 13–15.

[37] Barna Saha and Divesh Srivastava. 2014. Data quality: The other face of big data. In *IEEE 30th International Conference on Data Engineering, Chicago (ICDE'14).* 1294–1297.

[38] B. Saha and D. Srivastava. 2014. Data quality: The other face of Big Data. In *2014 IEEE 30th International Conference on Data Engineering*. 1294–1297.

[39] Mehdi Samadi, Partha Talukdar, Manuela Veloso, and Manuel Blum. 2016. ClaimEval: Integrated and flexible framework for claim evaluation using credibility of sources. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 222–228.

[40] Mehdi Samadi, Manuela M. Veloso, and Manuel Blum. 2013. OpenEval: Web information query evaluation. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI'13)*. AAAI Press, Bellevue, Washington, 1163–1169. http://dl.acm.org/citation.cfm?id=2891460.2891622.

[41] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, Christopher Potts, and others. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, Vol. 1631. Citeseer, 1642.

[42] Stephen Soderland, Oren Etzioni, Tal Shaked, and D. Weld. 2004. The use of web-based statistics to validate information extraction. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining*. 21–26.

[43] Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. 18–22.

[44] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. 2015. Quality assessment for linked data: A survey. *Semantic Web* 7, 1 (2015), 63–93.

[45] Xiaodan Zhu, Svetlana Kiritchenko, and Saif M. Mohammad. 2014. NRC-Canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval'14)*. Citeseer, 443–447.

[46] Valentin Zmiycharov, Dimitar Alexandrov, Preslav Nakov, Ivan Koychev, and Yasen Kiprov. 2017. Finding people's professions and nationalities using distant supervision: The FMI@SU "goosefoot" team at the WSDM Cup 2017 triple scoring task. In *WSDM Cup 2017 Notebook Papers, February 10, Cambridge, UK*, Martin Potthast, Stefan Heindorf, and Hannah Bast (Eds.). CEUR-WS.org. http://www.wsdm-cup-2017.org/proceedings.html.

Received April 2017; revised December 2017; accepted January 2018