DeFacto - Temporal and Multilingual Deep Fact Validation

Daniel Gerber, Diego Esteves*, Jens Lehmann**, Lorenz Bühmann, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, René Speck

University Leipzig, Institute for Computer Science, Agile Knowledge Engineering and Semantic Web (AKSW), D-04009 Leipzig, Germany

Abstract

One of the main tasks when creating and maintaining knowledge bases is to validate facts and provide sources for them in order to ensure correctness and traceability of the provided knowledge. So far, this task is often addressed by human curators in a three-step process: issuing appropriate keyword queries for the statement to check using standard search engines, retrieving potentially relevant documents and screening those documents for relevant content. The drawbacks of this process are manifold. Most importantly, it is very time-consuming as the experts have to carry out several search processes and must often read several documents. In this article, we present DeFacto (Deep Fact Validation) – an algorithm able to validate facts by finding trustworthy sources for them on the Web. DeFacto aims to provide an effective way of validating facts by supplying the user with relevant excerpts of web pages as well as useful additional information including a score for the confidence DeFacto has in the correctness of the input fact. To achieve this goal, DeFacto collects and combines evidence from web pages written in several languages. In addition, DeFacto provides support for facts with a temporal scope, i.e., it can estimate in which time frame a fact was valid. Given that the automatic evaluation of facts has not been paid much attention to so far, generic benchmarks for evaluating these frameworks were not previously available. We thus also present a generic evaluation framework for fact checking and make it publicly available.

Keywords: Web of Data, Fact Validation, NLP, Provenance

1. Introduction

The past decades have been marked by a change from an industrial society to an information and knowledge society. This change is particularly due to the uptake of the World Wide Web. Creating and managing knowledge successfully has been a key to success in various communities worldwide. Therefore, the quality of knowledge is of high importance. One aspect of knowledge quality is provenance (Zaveri et al., 2015). In particular, the sources for facts should be well documented since this provides several benefits such as a better detection of errors, decisions based on the trustworthiness of sources etc. While provenance is an important aspect of data quality (Hartig, 2009), to date only few knowledge bases actually provide provenance information. For instance, less than 10% of the more than 708.26 million RDF documents indexed by Sindice[1] contain metadata such as creator, creation date, source, modified or contributor.[2] This lack of provenance information makes the validation of the facts in such knowledge bases utterly tedious. In addition, it hinders the adoption of such data in business applications as the data is not trusted (Hartig, 2009). The main contribution of this paper is the provision of a fact validation approach and tool which can make use of one of the largest sources of information: the Web.

More specifically, our system DeFacto (Deep Fact Validation) implements algorithms for validating RDF triples by finding confirming sources for them on the Web.[3] It takes a statement as input (e.g., the one shown in Listing 1, page 13) and then tries to find evidence for the validity of that statement by searching for textual information in the Web. To this end, our approach combines two strategies by searching for textual occurrences of parts of the statements as well as trying to find web pages which contain the input statement expressed in natural language. DeFacto was conceived to exploit the multilinguality of the Web, as almost half of the content of the Web is written in a language other than English[4] (see Figure 1). To this end, our approach abstracts from a specific language and

*Corresponding author

**Principal corresponding author

Email addresses: `dgerber@informatik.uni-leipzig.de` (Daniel Gerber), `esteves@informatik.uni-leipzig.de` (Diego Esteves), `lehmann@informatik.uni-leipzig.de` (Jens Lehmann), `buehmann@informatik.uni-leipzig.de` (Lorenz Bühmann), `usbeck@informatik.uni-leipzig.de` (Ricardo Usbeck), `ngonga@informatik.uni-leipzig.de` (Axel-Cyrille Ngonga Ngomo), `speck@informatik.uni-leipzig.de` (René Speck)

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

[2] Data retrieved on February 13, 2015.

[3] Please note that we use *fact* as a synonym for a *RDF triple*.

[4] 45% non-English web pages according to `http://w3techs.com/technologies/overview/content_language/all`.

can combine evidence from multiple languages – currently English, German and French.

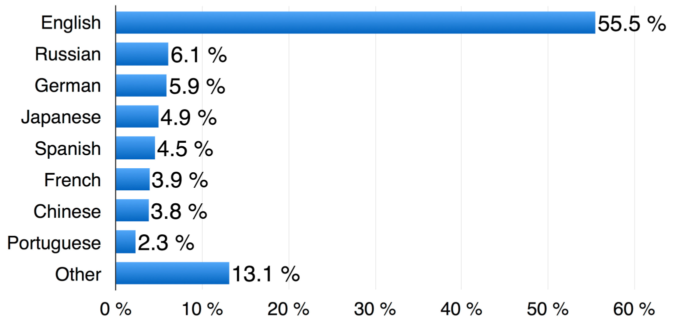


Figure 1: Usage of content languages for web pages. (W3Techs.com, 21 November 2013)

The output of our approach is a confidence score for the input statement as well as a set of excerpts of relevant web pages which allows the user to manually judge the presented evidence. Apart from the general confidence score, DeFacto also provides support for detecting the temporal scope of facts, i.e., estimates in which timeframe a fact is or was valid.

DeFacto has three major use cases: (1) Given an existing true statement, it can be used to find provenance information for it. For instance, the WikiData project[5] aims to create a collection of facts, in which sources should be provided for each fact. DeFacto could help to achieve this task. (2) It can check whether a statement is likely to be true and provide the user with a corresponding confidence score as well as evidence for the score assigned to the statement. (3) Given a fact, DeFacto can determine and present evidence for the time interval within which the said fact is to be considered valid. Our main contributions are thus as follows:

- We present an open-source approach that allows checking whether a web page confirms a fact, i.e., an RDF triple,
- We discuss an adaptation of existing approaches for determining indicators for trustworthiness of a web page,
- We present an automated approach to enhancing knowledge bases with RDF provenance data at triple level as well as
- We provide a running prototype of DeFacto, the first system able to provide useful confidence values for an input RDF triple given the Web as background text corpus.

This article is an extension of the initial description of DeFacto in (Lehmann et al., 2012). The main additions are as follows:

- A temporal extension detecting temporal scope of facts based on text understanding via pattern and frequency analysis.
- An extensive study of effect of the novel multilingual support in DeFacto, e.g., through the integration of search queries and temporal patterns in several languages.
- A freely available and full-fledged benchmark for fact validation which includes temporal scopes.

The rest of this paper is structured as follows: We give an overview of the state of the art of relevant scientific areas in Section 2. This part is followed by a description of the overall approach in a nutshell in Section 3. We show how we extended the BOA framework to enable it to detect facts contained in textual descriptions on web pages in Section 4. In Section 5, we describe how we calculate and include the trustworthiness of web pages into the DeFacto analysis. Section 6 combines the results from the previous chapters and describes the mathematical features we use to compute the confidence for a particular input fact. Subsequently, we describe the temporal extension of DeFacto in Section 7 and provide an overview of the FactBench benchmark in Section 8. We provide a discussion of the evaluation results in Section 9. Finally, we conclude in Section 10 and give pointers to future work.

2. Related Work

The work presented in this paper is related to five main areas of research: Fact checking as known from NLP, the representation of provenance information in the Web of Data, temporal analysis, relation extraction and named entity disambiguation (also called entity linking).

2.1. Fact Checking

Fact checking is a relatively new research area which focuses on computing which subset of a given set of statements can be trusted (Pasternack and Roth, 2013). Several approaches have been developed to achieve this goal. Nakamura et al. (2007) developed a prototype for enhancing the search results provided by a search engine based on trustworthiness analysis for those results. To this end, they conducted a survey in order to determine the frequency at which the users accesses search engines and how much they trust the content and ranking of search results. They defined several criteria for trustworthiness calculation of search results returned by the search engine, such as topic majority. We adapted their approach for DeFacto and included it as one of the features for our machine learning techniques. Another fact-finding approach is that presented in (Yin et al., 2007). Here, the idea is to create a 3-partite network of web pages, facts and objects and apply a propagation algorithm to compute weights for facts as well as web pages. These weights can then be used to

[5] http://www.wikidata.org

determine the degree to which a fact contained in a set of web pages can be trusted. Pasternack and Roth (2011a,b) present a generalized approach for computing the trustworthiness of web pages. To achieve this goal, the authors rely on a graph-based model similar to hubs and authorities (Kleinberg, 1999). This model allows computing the trustworthiness of facts and web pages by generating a k-partite network of pages and facts and propagating trustworthiness information across it. The approach returns a score for the trustworthiness of each fact. Moreover, the generalized fact-finding model that they present allows expressing other fact-finding algorithms such as TruthFinder (Yin et al., 2007), AccuVote (Dong et al., 2009) and 3-Estimates (Galland et al., 2010) within the same framework. The use of trustworthiness and uncertainty information on RDF data has been the subject of recent research (see e.g., (Hartig, 2008; Meiser et al., 2011)). Moreover, approaches such as random walks Jain and Pantel (2010) have been used to measure the trustworthiness of graph data based on the topology of the underlying graph. Our approach differs from previous fact finding works as it focuses on validating the trustworthiness of RDF triples (and not that of facts expressed in natural language) against the Web (in contrast to approaches that rely on the RDF graph only). In addition, it can deal with the broad spectrum of relations found on the Data Web.

2.2. Provenance

The problem of data provenance is an issue of central importance for the uptake of the Web of Data. While data extracted by the means of tools such as Hazy[6] and KnowItAll[7] can be easily mapped to primary provenance information, most knowledge sources were extracted from non-textual source and are more difficult to link with provenance information. Hartig and Zhao (2010) describes a framework for provenance tracking. This framework provides the vocabulary required for representing and accessing provenance information on the Web. It keeps track of metadata including who created a Web entity (e.g., a web page) and how the entity was modified. Recently, a W3C working group has been formed and released a set of specifications on sharing and representing provenance information.[8] Dividino et al. (2011) introduced an approach for managing several provenance dimensions, e.g., source, and timestamp. In their approach, they describe an extension to the RDF called RDF$^+$ which can work efficiently with provenance data. They also provide a method for enabling SPARQL query processors in a manner such that a specific SPARQL query can request meta knowledge without being modified. Theoharis et al. (2011) argue that the implicit provenance data contained in a SPARQL query result can be used to acquire annotations for several dimensions of data quality. They present the concept of abstract provenance models as known from databases and how it can be extended to suit the Data Web as well. DeFacto uses the W3C provenance group standard for representing provenance information. Yet, unlike previous work, it directly tries to find provenance information by searching for confirming facts in trustworthy web pages.

2.3. Temporal Analysis

Storing and managing the temporal validity of facts is a tedious task that has not yet been studied widely in literature. First works in from the Semantic Web community in this direction include Temporal RDF (Gutierrez et al., 2005), which allows representing time intervals within which a relation is valid. Extracting such information from structured data is a tedious endeavour for which only a small number of solutions exist. For example, (Talukdar et al., 2012b) present an approach for scoping temporal facts which relies on formal constraints between predicates. In particular, they make use of the alignment, containment, succession and mutual exclusion of predicates. Acquiring the constraints that hold between given predicates is studied in (Talukdar et al., 2012a). Another approach that aims at extracting temporal information is Timely YAGO (Wang et al., 2010), which focuses on extracting temporally scope facts from Wikipedia infoboxes. PRAVDA (Wang et al., 2011) relies on constrained label propagation to extract temporal information. Here, an objective function which models inclusion constraints and factual information is optimized to determine an assignment of fact to time slots. To the best of our knowledge, none of the previous approaches has dealt with coupling the validity of a fact with its time scope.

2.4. Relation Extraction (RE)

The verbalization of formal relations is an essential component of DeFacto as it allows searching for RDF triples in unstructured data sources. This verbalization task is strongly related to the area of relation extraction, which aims to detect formal relation relations between entity mentions in unstructured data sources. Some early work on relation extraction based on pattern extraction relied on supervised machine learning (see e.g., (Grishman and Yangarber, 1998)). Yet, such approaches demand large amounts of training data, making them difficult to adapt to new relations. The subsequent generation of approaches to RE aimed at bootstrapping patterns based on a small number of input patterns and instances. For example, Brin (1999) presents the Dual Iterative Pattern Relation Expansion (DIPRE) and applies it to the detection of relations between authors and titles of books. This approach relies on a small set of seed patterns to maximize the precision of the patterns for a given relation while minimizing their error rate of the same patterns. Snowball (Agichtein and Gravano, 2000) extends DIPRE by a new approach to the generation of seed tuples. Other

[6] `http://hazy.cs.wisc.edu/hazy/`

[7] `http://www.cs.washington.edu/research/knowitall/`

[8] `http://www.w3.org/2011/prov/wiki/`

approaches aim to either collect redundancy information (see e.g., (Yan et al., 2009)) in an unsupervised manner or to use linguistic analysis (Nguyen et al., 2007) to harvest generic patterns for relations. The latest approaches to relation extraction make use of ontologies as seed knowledge. While several approaches, including NELL (Carlson et al., 2010) and PROSPERA (Nakashole et al., 2011), use their own ontologies, frameworks such as BOA (Gerber and Ngonga Ngomo, 2012), LODifier (Augenstein et al., 2012) and DARE (Krause et al., 2012) reuse information available on the Linked Data Web as training data to discover natural-language patterns that express formal relations and reuse those to extract RDF from unstructured data sources.

2.5. Named Entity Disambiguation (NED)

NED is most often an a-priori task to RE. In the last years, approaches began relying on RDF data as underlying knowledge bases. *DBpedia Spotlight* (Mendes et al., 2011) is a Named Entity Recognition and Disambiguation combining approach based on DBpedia (Lehmann et al., 2009, 2014). This approach is able to work on all classes of Named Entities present in the knowledge base also enabling the user to specify coverage and error tolerance while the annotation task. Based on measures like prominence, topical relevance, contextual ambiguity and disambiguation conference DBpedia Spotlight achieves a disambiguation accuracy of 81% on their Wikipedia corpus. *AIDA* (Hoffart et al., 2011) is based on the YAGO[9] knowledge base. This approach uses dense sub-graphs to identify coherent mentions. Moreover, *AIDA* makes use of contextual similarity, prominence information and context windows. *AGDISTIS* (Usbeck et al., 2014) is a novel knowledge-base agnostic NED approach which combines an authority-based graph algorithm and different label expansion strategies and string similarity measures. Based on this combination, the approach can efficiently detect the correct URIs for a given set of named entities within an input text. The results indicate that *AGDISTIS* is able to outperform the state-of-the-art approaches by up to 16% F-measure.

3. Approach

Input and Output. The DeFacto system consists of the components depicted in Figure 2. It supports two types of inputs: RDF triples and textual data. If provided with a fact represented as an RDF triple as input, DeFacto returns a confidence value for this fact as well as possible evidence for it. In the case of textual data, e.g., from an input form, DeFacto disambiguates the entities and gathers surface forms (see Section 4) for each resource.[10] The evidence consists of a set of web pages, textual excerpts from those pages and meta-information on the pages. The text excerpts and the associated meta information enable the user to quickly obtain an overview of possible credible sources for the input statement. Instead of having to use search engines, browsing several web pages and looking for relevant pieces of information, the user can thus more efficiently review the presented information. The system uses techniques which were adapted specifically for fact validation rather than relying only on generic information retrieval techniques of search engines.

Retrieving Web Pages. The first step of the DeFacto fact validation process is to retrieve web pages which are relevant for the given task. The retrieval is carried out by issuing several queries to a regular search engine. These queries are computed by verbalizing the fact using multilingual natural-language patterns extracted by the BOA framework[11] (Gerber and Ngonga Ngomo, 2011, 2012). Section 4.2 describes how the search engine queries are constructed. In a subsequent step, the highest ranked web pages for each query are retrieved. Those web pages are candidates for being evidence sources for the input fact. Both the search engine queries as well as the retrieval of web pages are executed in parallel to keep the response time for users within a reasonable limit. Note, that usually this does not put a high load on particular web servers as web pages are usually derived from several domains.

Evaluating Web Pages. Once all web pages have been retrieved, they undergo several further processing steps. First, plain text is extracted from each web page by removing most HTML markup. We can then apply our fact confirmation approach on this text, which is described in detail in Section 4.3. In essence, the algorithm decides whether the web page contains a natural language formulation of the input fact. This step distinguishes DeFacto from information retrieval methods. If no web page confirms a fact according to DeFacto, then the system falls back on lightweight NLP techniques and computes whether the web page does at least provide useful evidence. In addition to fact confirmation, the system computes different indicators for the trustworthiness of a web page (see Section 5). These indicators are of central importance because a single trustworthy web page confirming a fact may be a more useful source than several web pages with low trustworthiness. The fact confirmation and the trustworthiness indicators of the most relevant web pages are presented to the user.

Confidence Measurement. In addition to finding and displaying useful sources, DeFacto also outputs a general confidence value for the input fact. This confidence value ranges between 0% and 100% and serves as an indicator for the user: Higher values indicate that the found sources

[9]`http://www.mpi-inf.mpg.de/yago-naga/yago/`

[10]Note the disambiguation of the property URI of the fact is out of scope of this paper.

[11]`http://boa.aksw.org`

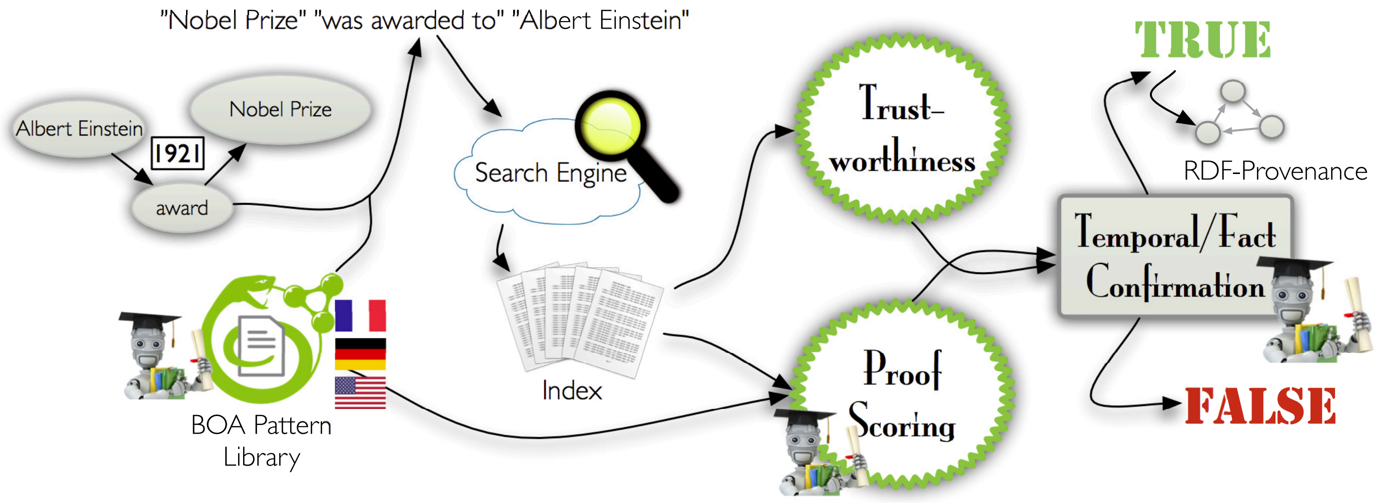


Figure 2: Overview of the DeFacto Architecture.

appear to confirm the fact and can be trusted. Low values mean that not much evidence for the fact could be found on the Web and that the web pages that do confirm the fact (if such exist) only display low trustworthiness. The confidence measurement is based on machine learning techniques and explained in detail in Sections 6 and 9. Naturally, DeFacto is a (semi-)automatic approach: We do assume that users will not blindly trust the system, but additionally analyze the provided evidence.

RDF Provenance Output.
Besides a visual representation of the fact and its most relevant web pages, it is also possible to export this information as RDF, which enables a Linked Data style access and/or storing in a SPARQL endpoint. We reuse several existing vocabularies for modeling the provenance of the DeFacto output (see Figure 3), especially the PROV Ontology (Belhajjame et al., 2012), which provides a set of classes, properties, and restrictions that can be used to represent and interchange provenance information generated in different systems and under different contexts, and the Natural Language Processing Interchange Format (NIF) (Hellmann et al., 2013), which is an RDF/OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources and annotations. A RDF dump of the generated evidences for the correct facts of FactBench (see Section 8) can be downloaded from the project home page[12].

DeFacto Web Demo.
A prototype implementing the above steps is available at `http://defacto.aksw.org`. A screenshot of the user interface is depicted in Figure 4. It shows relevant web pages, text excerpts and five different rankings per page. As described above, the generated provenance output can also be saved directly as RDF using the W3C provenance group[13] vocabularies. The source code of both the DeFacto algorithms and user interface are openly available.[14]

It should be noted that we decided not to check for negative evidence of facts in DeFacto, since a) we considered this to be too error-prone and b) negative statements are much less frequent on the Web.

4. BOA - Bootstrapping Linked Data

The idea behind BOA is two-fold: First, it aims to be a framework that allows extracting structured data from the Human-readable Web by using Linked Data as background knowledge. In addition, it provides a library of natural-language patterns for formal relations that allows bridging the gap between structured and unstructured data. The input for the BOA framework consists of a set of knowledge bases, a text corpus (mostly extracted from the Web) and (optionally) a Wikipedia dump[15]. When provided with a Wikipedia dump, the framework begins by generating surface forms for all entities in the source knowledge base. The surface forms used by BOA are generated by using an extension of the method proposed in (Mendes et al., 2011). For each predicate p found in the input knowledge sources, BOA carries out a sentence-level statistical analysis of the co-occurrence of pairs of labels of resources that are linked via p. BOA then uses a supervised machine-learning approach to compute a score and rank patterns for each combination of corpus and knowledge bases. These patterns allow generating a natural-language representation (NLR) of the RDF triple that is to be checked.

[12]`http://aksw.org/Projects/DeFacto`

[13]`http://www.w3.org/2011/prov/`
[14]`https://github.com/AKSW/DeFacto`
[15]`http://dumps.wikimedia.org/`

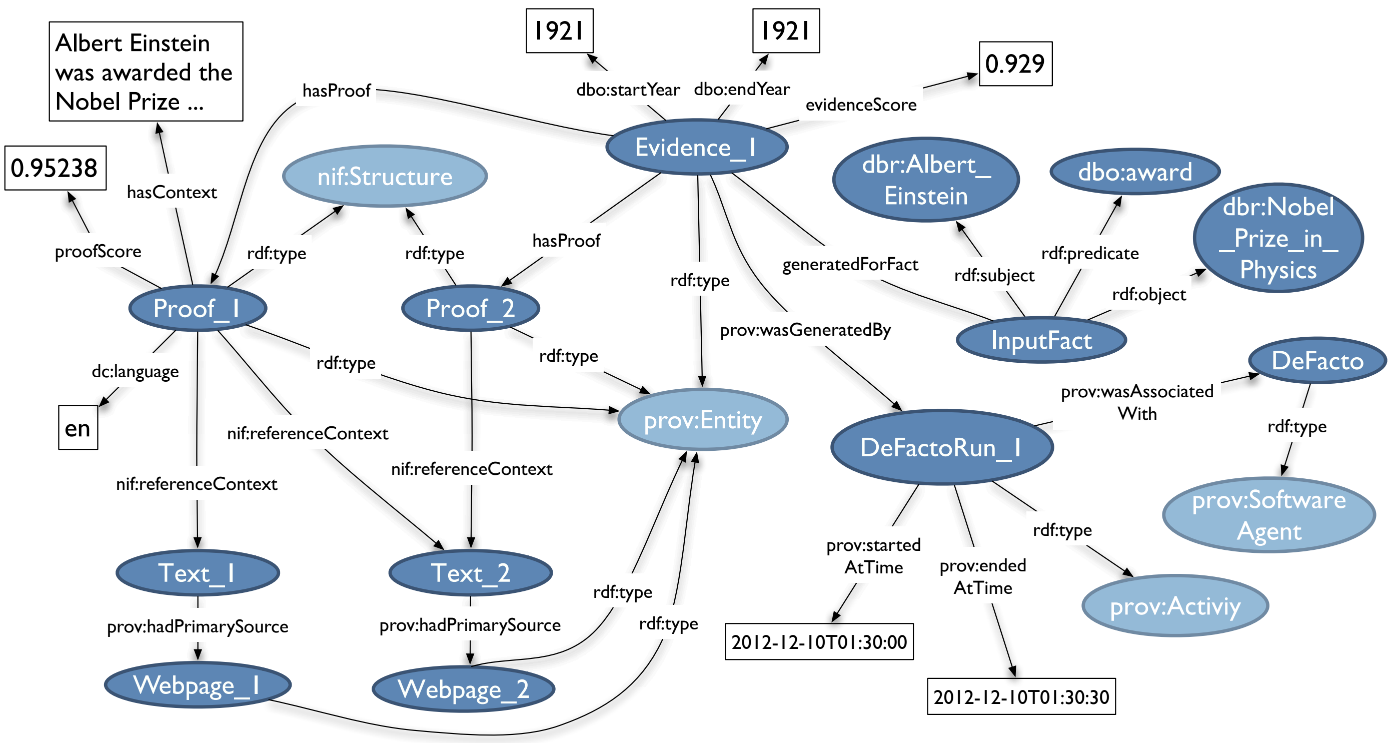


Figure 3: Overview of the provenance schema which is used to export the validation result of DeFacto as RDF, given the input fact `Albert Einstein, award, Nobel Price in Physics`.

4.1. Training BOA for DeFacto

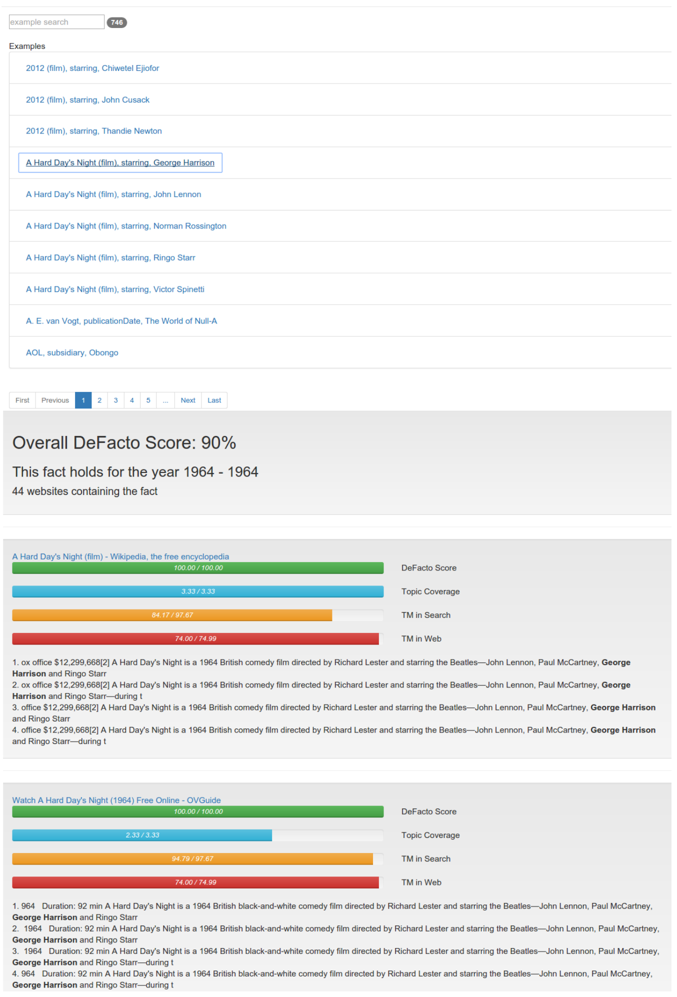
In order to provide a high quality fact confirmation component, we trained BOA specifically for this task. We began by selecting the relations used in FactBench (see Section 8) and queried the instance knowledge from DBpedia 3.9 (see (Lehmann et al., 2009, 2014)). Since first experiments showed that relying on the localized version of DBpedia would result in poor recall, we translated the English background knowledge to German and French respectively. This is carried out by replacing English *rdfs:labels* with localized ones if such exists. If no target language label exists, we rely on the English label as backup. We then ran BOA on the July 2013 dumps of the corresponding Wikipedias. Since search engine queries are expensive we ordered the generated patterns by their support set size, the subject and object pairs the patterns was found from, and used the top-n patterns for each relation to formulate search engine queries. We chose not to train BOA's machine learning module, since this would have resulted in high-precision but low-recall patterns. Additionally, we implemented a pattern generalization approach to better cope with similar but low-recall patterns.

Overall, all components of DeFacto can be trained so as to be used a domain different than the domains of DBpedia. If no evidence for a fact is available on the Web, then DeFacto will be unable to determine the validity of the corresponding fact. One approach towards still being able to determine the validity of a fact would then be to provide DeFacto with a specialized corpus that contains information pertaining to the resources involved in the fact.

The figure 5 shows the component diagram for DeFacto, which implements a component-modularized architecture in order to aid library extensions as easy as possible. To achieve a higher level of decoupling, the implementation of interfaces is planned as future work. Further, in spite of its modularization for the purpose of use it in any relation domain, DeFacto should be adapted to work in knowledge base presenting new relations. This can be attained by extracting the new set of existing patterns (*FactBench component*) from given data source having new relations that were not covered so far. The Fact Confirmation classifier, derived from patterns generated from the DBPedia by BOA (along which the Fact Bench), could be obtained from different knowledge bases by re-training the algorithm (FactBench component). However, an adoption of existing thresholds and measures in order to optimize the model and avoid overfitted models is needed.

A further analysis of the variation and quality aspect of patterns extracted in different datasources is desired as future work.

Lexical Pattern Generalization for DeFacto. A drawback of the previous version of the BOA framework was that it could not detect similar patterns. For example consider the following two English patterns: "?R 's Indian subsidiary ?D" and "?R 's UK subsidiary , ?D". Both patterns are NLRs for the *dbo:subsidiary* relation but might


example search 746

Examples

2012 (film), starring, Chiwetel Ejiofor

2012 (film), starring, John Cusack

2012 (film), starring, Thandie Newton

A Hard Day's Night (film), starring, George Harrison

A Hard Day's Night (film), starring, John Lennon

A Hard Day's Night (film), starring, Norman Rossington

A Hard Day's Night (film), starring, Ringo Starr

A Hard Day's Night (film), starring, Victor Spinetti

A. E. van Vogt, publicationDate, The World of Null-A

AOL, subsidiary, Obongo

First Previous 1 2 3 4 5 ... Next Last

Overall DeFacto Score: 90%

This fact holds for the year 1964 - 1964

44 websites containing the fact

A Hard Day's Night (film) - Wikipedia, the free encyclopedia

100.00 / 100.00	DeFacto Score
3.33 / 3.33	Topic Coverage
84.17 / 97.67	TM in Search
74.00 / 74.99	TM in Web

1. ox office $12,299,668[2] A Hard Day's Night is a 1964 British comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr
2. ox office $12,299,668[2] A Hard Day's Night is a 1964 British comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr—during t
3. office $12,299,668[2] A Hard Day's Night is a 1964 British comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr
4. office $12,299,668[2] A Hard Day's Night is a 1964 British comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr—during t

Watch A Hard Day's Night (1964) Free Online - OVGuide

100.00 / 100.00	DeFacto Score
2.33 / 3.33	Topic Coverage
94.79 / 97.67	TM in Search
74.00 / 74.99	TM in Web

1. 964 Duration: 92 min A Hard Day's Night is a 1964 British black-and-white comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr
2. 1964 Duration: 92 min A Hard Day's Night is a 1964 British black-and-white comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr
3. 1964 Duration: 92 min A Hard Day's Night is a 1964 British black-and-white comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr—during t
4. 964 Duration: 92 min A Hard Day's Night is a 1964 British black-and-white comedy film directed by Richard Lester and starring the Beatles—John Lennon, Paul McCartney, **George Harrison** and Ringo Starr—during t

Figure 4: Screenshot of the DeFacto Web interface.

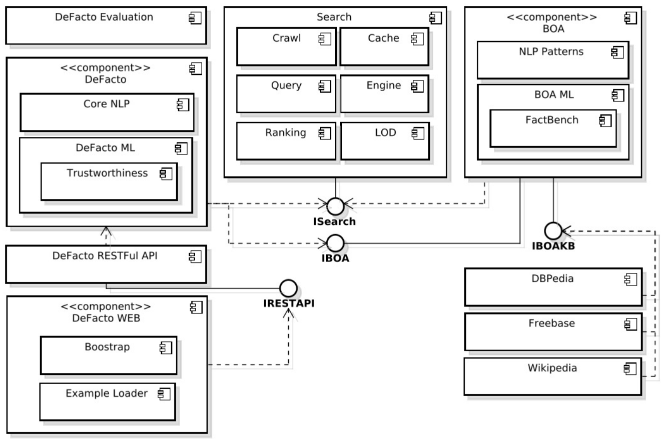


Figure 5: The architecture of the main services represented on the component diagram

fail to score high confidence scores because of their individual low number of occurrences. Generalizing these patterns into "?R 's NE subsidiary ?D" can therefore help boost pattern scores for low recall patterns. We generalize patterns individually for each language based on manually crafted regular expressions and Part-Of-Speech(POS) tags provided by a language-specific tagger. In the current version of DeFacto, we generalize personal pronouns, named entities, date/year occurrences and forms of "be" as well as numerical values.

4.2. Automatic Generation of Search Queries

The found BOA patterns are used for issuing queries to the search engine (see Figure 2). Each search query contains the quoted label (forces an exact match from the search engine) of the subject of the input triple, a quoted and cleaned BOA pattern (i.e., without punctuation) and the quoted label of the object of the input triple. Note that we can fully localize the search query in most cases since there are multi-lingual labels for many resources available on the LOD cloud. We use the top-k best-scored BOA patterns and retrieve the first n web pages from a Web search engine[16]. For our example from Listing 1, an exemplary query sent to the search engine is as follows:

``"Albert Einstein'' AND ``was awarded the'' AND ``Nobel Prize in Physics''.``

We then crawl each web page, remove HTML markup and try to extract *possible proofs* for the input triple, i.e., excerpts of these web pages which may confirm it. For the sake of brevity, we use proof and possible proof interchangeably.

4.3. BOA and NLP Techniques for Fact Confirmation

To find proofs for a given input triple $t = (s, p, o)$ we make use of the surface forms introduced in (Mendes et al., 2011). We select all surface forms for the subject and object of the input triple and search for all occurrences of each combination of those labels in a web page w. If we find an occurrence with a token distance $dist(l(s), l(o))$ (where $l(x)$ is the label of x in any of the configured languages) smaller then a given threshold we call this occurrence a proof for the input triple. To remove noise from the found proofs we apply a set of normalizations by using regular expression filters which for example remove characters between brackets and non alpha-numeric characters. Note that this normalization improves the grouping of proofs by their occurrence. After extracting all proofs $pr_i \in \mathcal{P}r_w$ of a web page w, we score each proof using a logistic regression classifier (Landwehr et al., 2005). We trained a classifier with the following input features for scoring a proof:

String Similarity For the top-n BOA patterns of the given relation we determine the maximum string similarity between the normalized pattern and the proof phrase. As string similarity we use Levenshtein, QGram Similarity as well as Smith-Waterman.[17]

Token Distance: This is the distance $dist(l(s), l(o))$ between the two entity labels which found the proof. We limit this distance to a maximum of 10 tokens.

[16] Bing Web Search and Google Web Search
[17] `http://sourceforge.net/projects/simmetrics/`

	$en_{20\%}$	$en_{100\%}$	$de_{20\%}$	$de_{100\%}$	$fr_{20\%}$	$fr_{100\%}$
True	12414	79921	4419	29292	5724	36383
False	11705	17436	5488	8263	5231	7721
Total	24119	97357	9907	37555	10955	44104

Table 1: Proofs with language distribution used to train fact classifier.

Wordnet Expansion: We expand both the tokens of the normalized proof phrase as well as all of the tokens of the BOA pattern with synsets from Wordnet. Subsequently we apply the Jaccard-Similarity on the generated expansions. This is basically a fuzzy match between the BOA pattern and the proof phrase. Due to the language specificity of Wordnet to English, we will use BabelNet (see (Navigli and Ponzetto, 2012)) in future iterations.

Syntax: We also calculate a number of numeric features for the proof phrases: the number of uppercase and non-alpha-numeric characters, the number of commas, digits and characters and the average token length.

Total Occurrence: This feature contains the total number of occurrences of each normalized proof phrase over the set of all normalized proof phrases.

Page Title: We calculate the maximum of the Levenshtein similarity between the page title and the subject and object labels. This feature is useful, because the title indicates the topic of the entire web page. When a title matches, then higher token distances may still indicate a high probability that a fact is confirmed.

End of Sentence: The number of occurrences of ".", "!" or a "?" in the proof context. When subject and object are in different sentences, their relation is more likely to be weaker.

Proof Phrase: The words in the proof phrase between subject and object, which are encoded as binary values, i.e., a feature is created for each word and its value is set to 1 if the word occurs and 0 otherwise.

Property: The property as a word vector.

Language: The language of the web page.

4.4. DeFacto Training

To train our classifier, we ran DeFacto on the *mix* train set (see Section 8) and extracted all proof phrases. We randomly sampled 20% of the 178337 proofs, trained the classifier on 66.6% and evaluated the learned model on the 33.3% unseen proofs. Both the train and the test set contained an equal amount of instances of both classes. A detailed overview of the proofs used to learn the fact classifier can be seen in table 4.3. As expected, there is a skew towards proofs extracted in English (2.4 for English to German, 2.2 for English to French). This is not surprising, since English is the dominant language on the Web (see Figure 1). We chose an SVM as classifier since it is known to be able to handle large sets of features[18] and is able to work with numeric data and create confidence values. The ability to generate confidence values for proofs is useful as feedback for users and it also serves as input for the core classifiers described in Section 6. We achieved an F_1 score of 74.1%. We also performed preliminary work on fine-tuning the parameters of the above algorithms, which, however, did not lead to significantly different results. Therefore, the reported measurements were carried out with default values of the mentioned algorithms in the Weka machine learning toolkit[19] version 3.6.6.

5. Trustworthiness Analysis of Web Pages

To determine the trustworthiness of a web page, we first determine its similarity to the input triple – usually pages on topic related to the input triple are more valuable. This is determined by how many topics belonging to the query are contained in a search result retrieved by the web search. We extended the approach introduced in (Nakamura et al., 2007) by querying Wikipedia with the subject and object label of the triple in question separately to find the topic terms for the triple. Please note that through the availability of multi-lingual labels for many resources in the LOD cloud, we are able to extract topic terms in multiple languages. A frequency analysis is applied on all returned documents and all terms above a certain threshold that are not contained in a self-compiled stop word list are considered to be topic terms for a triple. Let s and o be the URIs for the subject and object of the triple in question, τ be a potential topic term extracted from a Wikipedia page and let $t = (s, p, o)$ be the input triple. We compare the values of the following two formulas:

$$prob(\tau|t) = \frac{|topic(\tau, docs(t))|}{|docs(t)|} \tag{1}$$

$$prob(t|intitle(docs(t), s \vee o)) = \frac{|topic(\tau, intitle(docs(t), s) \cup intitle(docs(t), o))|}{|intitle(docs(t), s) \cup intitle(docs(t), o)|} \tag{2}$$

where $docs(t)$ is the set all web documents retrieved for t (see Section 4.2), $intitle(docs(t), x)$ the set of web documents which have the label of the URI x in their page title.

[18]Note that the majority of the features are word vectors.
[19]`http://www.cs.waikato.ac.nz/ml/weka/`

	English	German	French
publication	?R 's novel " ?D	?R in seinem Roman " ?D	?D " est un roman ?R
	?R 's book " ?D	?R in seinem Buch " ?D	?R dans son roman " ?D
	?R , author of " ?D	?R in seinem Werk " ?D	?R intitulé " ?D
marriage	?R married ?D	?D seiner Frau ?R	?R épouse ?D
	?R , his wife ?D	?D seiner Ehefrau ?R	?R , veuve ?D
	?D 's marriage to ?R	?R und seiner Gattin ?D	?D , la femme de ?R

Table 2: Example list of patterns for relations publication and marriage.

$topic(\tau, docs(t))$ is a function returning the set of documents which contain τ in the page body. We consider τ to be a topic term for the input triple if $prob(\tau|\tau(docs(t), s) \vee \tau(docs(t), o)) > prob(\tau|t)$. Let $\mathfrak{T}_t = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be the set of all topic terms extracted for an input triple. DeFacto then calculates the trustworthiness of a web page as follows:

Topic Majority in the Web. This represents the number of web pages that have similar topics to the web page in question. Let $\mathfrak{T}_w$ be the set of topic terms appearing on the current web page w. The Topic Majority in the Web for a web page w is then calculated as:

$$tm_{web}(w) = \left| \bigcup_{i=1}^{n} topic(\tau_i, d(X)) \right| - 1. \qquad (3)$$

where τ_1 is the most frequently occurring topic term in the web page w. Note that we subtract 1 to prevent counting w.

Topic Majority in Search Results. This is used to calculate the similarity of a given web page to all web pages found for a given triple. Let w_k be the web page to be evaluated, $v(w_k)$ be the feature vector of web page w_k where $v(w_k)_i$ is 1 if τ_i is a topic term of web page w_k and 0 otherwise, $\|v\|$ be the norm of v and θ a similarity threshold. We calculate the Topic Majority in Search Results as follows:

$$tm_{sr}(w) = \left| \left\{ w_i | w_i \in d(X), \frac{v(w_k) \times v(w_i)}{\|v(w_k)\| \, \|v(w_i)\|} > \theta \right\} \right|. \qquad (4)$$

Topic Coverage. This measures the ratio between all topic terms for t and all topic terms occurring in w:

$$tc(w) = \frac{|\mathfrak{T}_t \cap \mathfrak{T}_w|}{|\mathfrak{T}_t|}. \qquad (5)$$

6. Features for Deep Fact Validation

In order to obtain an estimate of the confidence that there is sufficient evidence to consider the input triple to be true, we chose to train a supervised machine learning algorithm. Similar to the above presented classifier for fact confirmation, this classifier also requires computing a set of relevant features for the given task. In the following, we describe those features and why we selected them.

First, we extend the score of single proofs to a score of web pages as follows: When interpreting the score of a proof as the probability that a proof actually confirms the input fact, then we can compute the probability that at least one of the proofs confirms the fact. This leads to the following stochastic formula[20], which allows us to obtain an overall score for proofs scw on a web page w:

$$scw(w) = 1 - \prod_{pr \in prw(w)} (1 - fc(pr)). \qquad (6)$$

In this formula, fc (fact confirmation) is the classifier trained in Section 4.3, which takes a proof pr as input and returns a value between 0 and 1. prw is a function taking a web page as input and returning all possible proofs contained in it.

Combination of Trustworthiness and Textual Evidence. In general, we assume that the trustworthiness of a web page and the textual evidence found in it are orthogonal features. Naturally, web pages with high trustworthiness and a high score for its proofs should increase our confidence in the input fact. We thus combine trustworthiness and textual evidence as features for the underlying machine learning algorithm. This is achieved by multiplying both criteria and then using their sum and maximum as two different features:

$$F_{fsum}(t) = \sum_{w \in s(t)} (f(w) \cdot scw(w)) \qquad (7)$$

$$F_{fmax}(t) = \max_{w \in s(t)} (f(w) \cdot scw(w)) \qquad (8)$$

In this formula, f can be instantiated by all three trustworthiness measures: topic majority on the the Web (tm_{web}), topic majority in search results (tm_{sr}) and topic coverage (tc). s is a function taking a triple t as argument, executing the search queries explained in Section 4.2 and returning a set of web pages. Using the formula, we obtain 6 different features for our classifier, which combine textual evidence and different trustworthiness measures.

[20]To be exact, it is the complementary even to the case that none of the proofs do actually confirm a fact.

Other Features. In addition to the above described combinations of trustworthiness and fact confirmation, we also defined other features:

1. The **total number of proofs** found.
2. The **total number of proofs found above a relevance threshold** of 0.5. In some cases, a high number of proofs with low scores is generated, so the number of high scoring proofs may be a relevant feature for learning algorithms. The thresholds mimics a simple classifier.
3. The **total evidence score**, i.e., the probability that at least one of the proofs is correct, which is defined analogously to *scw* above:
$$1 - \prod_{pr \in prt(t)} (1 - fc(pr)). \qquad (9)$$
where $prt(t)$ is a function returning all proofs found for t from all web pages.
4. The **total evidence score above a relevance threshold** of 0.5. This is an adaption of the above formula, which considers only proofs with a confidence higher than 0.5.
5. The **total hit count**, i.e., search engine's estimate of the number of search results for an input query. The total hit count is the sum of the estimated number of search results for each query send by DeFacto for a given input triple.
6. A **domain and range verification**: If the subject of the input triple is not an instance of the domain of the property of the input triple, this violates the underlying schema, which should result in a lower confidence in the correctness of the triple. This feature is 0 if both domain and range are violated, 0.5 if exactly one of them is violated and 1 if there is no domain or range violation. At the moment, we are only checking whether the instance is asserted to be an instance of a class (or one of its subclasses) and do not use reasoning for performance reasons.
7. **Statistical triple evidence**: Usually certain classes have a higher probability to cooccur as type of subject and object in a given triple, e.g., there might be a higher probability that instances of `dbo:Person` and `dbo:Film` are related via triples than for instance `dbo:Insect` and `dbo:Film`. This observation also holds for the cooccurence of classes and properties, both for the types in subject and object position. This kind of semantic relatedness allows for computing a score for the statistical evidence STE of a triple $t = (s, p, o)$ by
$$STE(t) = \max_{\substack{c_s \in cls(s) \\ o_s \in cls(o)}} (PMI(c_s, c_o) + PMI(c_s, p) + PMI(p, c_o)) \qquad (10)$$
where cls denotes the types of the resource and PMI denotes the Pointwise Mutual Information, which is a measure of association and defined by
$$PMI(a, b) = log\left(N \cdot \frac{occ(a, b)}{occ(a) \cdot occ(b)}\right) \qquad (11)$$
using $occ(e)$ as number of occurrences of a given entity e in a specific position of a triple and N as the total number of triples in the knowledge base.

7. Temporal Extension of DeFacto

A major drawback of the previous version of DeFacto was the missing support of temporal validation. There was no way to check if a triple, e.g., `<Tom_Cruise> <spouse <Katie_Holmes>`, is still true or if it only has been valid in the past. To overcome this drawback we introduce a temporal extension of DeFacto which is able to handle facts that happened on a particular date (*time points*) and facts which span a longer duration (*time periods*). The granularity of both time points and time periods is years. In this section we describe the two sources for determining the correct time point/period: *a*) statistics, e.g., the (normalized) frequencies of years in proof phrases; and *b*) the combination of statistics and text understanding, by finding lexical patterns expressing temporal information in free text.

7.1. Temporal Pattern Extraction

Our temporal pattern extraction method takes a set of corpora as input. Each corpus is split and indexed on sentence level. After that, we perform index lookups for all sentences which contain at least one of all possible combinations of two years between 1900 and 2013. We apply a context window around the year match in the sentence which ensures that all text excerpts contain at least three (if possible) tokens before, between and after the two year occurrences. Furthermore, we replaced the year occurrences with placeholders and created a frequency distribution for all matching patterns. We then manually relaxed the patterns by generating regular expression versions and added additional ones by examining the occurrences of the years in the DeFacto training set.

Specifically in our experiment, we used the Wikipedia dumps (generated in July 2013) as text corpora for the pattern search. After splitting the articles in sentences the English index contained 64.7M sentences where as the German with 27.6M and the French with 17.0M sentences are significantly smaller. An excerpt of the used patterns can be seen in Table 3 and the complete list of all patterns can be downloaded at the project home page.

7.2. Year Frequency

We apply two different year extraction mechanisms for facts with associated time points and time periods. In the first case, we create a frequency distribution for tokens

```
[0-9]{4}\\s*(/|-|--|-)\\s*[0-9]{4}
[Ff]rom [0-9]{4} until [0-9]{4}
[bB]etween (the years) [0-9]{4} and [0-9]{4}
[0-9]{4} bis einschließlich [0-9]{4}
[zZ]wischen (den Jahren) [0-9]{4} und [0-9]{4}
[dD]urant la période [0-9]{4} - [0-9]{4}
[eE]ntre les années [0-9]{4} et [0-9]{4}
```

Table 3: Example list of temporal patterns extracted from English, German and French Wikipedia.

which match the regular expression "[1-2][0-9]{3}". To this end, we take all complex proof phrases extracted from DeFacto for a given triple and create a context window (of variable length) to the left and right of the surface form of the fact's subject and object. All tokens from all context windows for all complex proofs are combined to compute the year frequency for an input fact. In the case of an associated time period we first try to find year pairs. To find those year pairs we extract a list of multilingual temporal regular expression patterns (see Subsection 7.1). We apply these patterns to the context of all complex proofs and create a frequency distribution for the start and end year of a given time period. We then choose the most frequent years from both distributions as start- or end-point of the given fact. In case the temporal patterns do not return any year pairs, we apply a frequency analysis as explained for time points and select the first two years as start or end.

7.3. Normalizing Frequencies of Years

The extraction of time information is strongly influenced by the growing amount of digital content over time. This leads to more recent dates being more likely to be found on the Web, as shown in Figure 6, than less recent ones. Within DeFacto, we thus implemented two different approaches for normalizing the frequency of years by their popularity on the Web. Each approach defined a *popularity function pop*, which takes a year as input and returns a number representing its popularity. Values below 1 indicate that the year has less than average popularity, where values above 1 indicate an above average popularity. When collecting evidence in DeFacto, we can then divide the frequency of all years found by their popularity value to obtain a distribution, which takes popularity into account. When two years are equally often associated to a particular event, this means DeFacto will assign a higher probability to the year with lower *pop*-value.

Global Normalization. Using the data from Figure 6, we have an estimate of the frequency of years, or more specifically a set Y of 4-digit-numbers, on the Web. Assuming that wf (Web frequency) is a function taking a year as input and returning its frequency on the Web, we can define the popularity function as follows:

$$pop_{global}(x) = \sqrt{\frac{wf(x)}{\frac{1}{|Y|}\sum_{y\in Y} wf(y)}} \qquad (12)$$

This function divides the frequency of a year on the Web by the average frequency of all years in Y on the Web. The square root is used to soften the effect of the normalization.

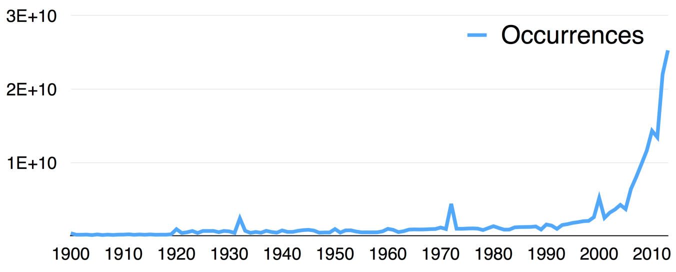


Figure 6: Distribution of year numbers in World Wide Web. Shows approximate number of Google search results. Outliers from left to right, 1931, 1972 and 2000. As comparison 'EU' has about 2.280.000.000 and 'Obama' 478.000.000 hits.

Domain-Specific Normalization. A second option to compute the popularity value of a year is to use the training set and actual evidence obtained by DeFacto. In Section 4, we described how search queries are generated and text excerpts (proofs) are extracted from the resulting web pages. Let pf (proof frequency) be a function taking a year as input and returning its frequencies in all proofs generated when running DeFacto over a training set. Furthermore, let tf (training set frequency) be the frequency of correct years in the training set.

Intuitively, we can expect years frequently in the training set to also frequently occur in the generated proofs. Therefore, we first divide pf by tf and then apply an analogous approach to the global normalization introduced above:

$$pop_{domain}(x) = \sqrt{\frac{\frac{pf(x)}{tf(x)}}{\frac{1}{|Y|}\sum_{y\in Y}\frac{pf(y)}{tf(y)}}} \qquad (13)$$

The fictional example below shows the results of our approach when using only three years as input:

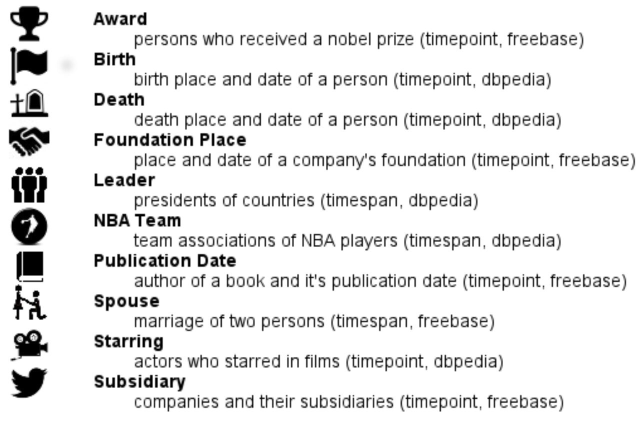
	1990	2000	2010
pf	30	30	100
tf	5	3	3
pop_{domain}	0.36	0.61	2.03

In this example, 2000 is more popular than 1990, since it has the same frequency in proofs, but a lower frequency in the training set. 2010 is more popular than 2000, since it has a higher frequency in proofs, but the same frequency in the training set.

8. FactBench - A Fact Validation Benchmark

FactBench is a multilingual benchmark for the evaluation of fact validation algorithms. All facts in FactBench are scoped with a timespan in which they were true, enabling the validation of temporal relation extraction algorithms. FactBench currently supports English, German and French. The current release V1 is freely available (MIT License) at `http://github.com/AKSW/FactBench`. FactBench consists of a set of RDF models. Each one of the 1500 models contains a singular fact and the time period in which it holds true. Each fact was checked manually by three independent human quality raters. In addition, the FactBench suite contains the SPARQL and MQL queries used to query Freebase[21] and DBpedia, a list of surface forms for English, French and German as well as the number of incoming and outgoing links for the English wikipedia pages. FactBench provides data for 10 well-known relations. The data was automatically extracted from DBpedia and Freebase. A detailed description on what facts the benchmark contains is shown in Figure 7. The granularity of FactBench's time information is year. This means that a timespan is an interval of two years, e.g., 2008 - 2012. A time point is considered as a timespan with the same start and end year, e.g., 2008 - 2008.

FactBench is divided in a training and a testing set (of facts). This strict separation avoids the overfitting of machine learning algorithms to the training set, by providing unseen test instances.



Award
persons who received a nobel prize (timepoint, freebase)
Birth
birth place and date of a person (timepoint, dbpedia)
Death
death place and date of a person (timepoint, dbpedia)
Foundation Place
place and date of a company's foundation (timepoint, freebase)
Leader
presidents of countries (timespan, dbpedia)
NBA Team
team associations of NBA players (timespan, dbpedia)
Publication Date
author of a book and it's publication date (timepoint, freebase)
Spouse
marriage of two persons (timespan, freebase)
Starring
actors who starred in films (timepoint, dbpedia)
Subsidiary
companies and their subsidiaries (timepoint, freebase)

Figure 7: FactBench provides data for 10 relations. The data was automatically extracted from Wikipedia (DBpedia respectivly) and Freebase

8.1. Expression of Temporal Information in RDF Knowledge Bases

There are several methods to model temporal information in the Web of Data. According to Rula et al. (2012), we can distinguish the following main categories:

- Document-centric, e.g., time points are connected to documents via the last modified HTTP header.
- Fact-centric, e.g., temporal information refers to facts. This can be divided into sentence-centric and relationship centric perspectives. In the sentence-centric perspective, the temporal validity of one or more statements is defined by annotating the facets. In the relationship centric perspective, n-ary relations are used to encapsulate temporal information.

Popular knowledge bases show a variety of different modeling choices for temporal information: DBpedia uses a class `dbo:TimePeriod` and attaches various properties to it, e.g., `dbo:activeYearsEndDate` is used to associate an `xsd:date` to mark the end of some activity. Freebase is fact-centric, more specifically relationship centric, and uses n-ary relations. YAGO is sentence-centric and uses reification to attach temporal restrictions to statements. Furthermore, there exist a variety of ontologies and standards related to representing temporal information, e.g., the OWL time ontology[22], XML Schema Date Datatypes[23], ISO standard 8601[24], Dublin Core time interval encoding[25] and Linked Timelines (Correndo et al., 2010). For FactBench, we adopt a temporal representation similar to the one used in DBpedia, although other options appear to be equally appropriate. Listing 1 shows an example fact.

8.2. Data Generation

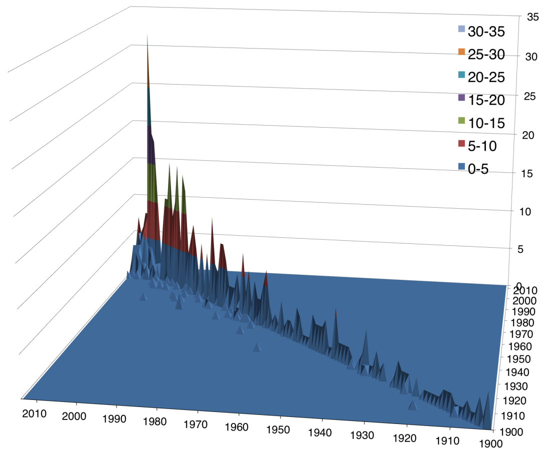


Figure 8: Overview of time points and time periods in the FactBench train set.

The data generation of the FactBench benchmark has three objectives. (1) We try to cover as many different domains as possible. The benchmark includes, amongst others, relations from the persons (marriage), places (birth,

[21]Since there are no incremental releases from Freebase we include the crawled training data

[22]`http://www.w3.org/TR/owl-time/`
[23]`www.w3.org/TR/xmlschema-2/`
[24]`http://en.wikipedia.org/wiki/ISO_8601#Time_intervals`
[25]`http://dublincore.org/documents/dcmi-period/`

death) and organizations domain (subsidiary). In addition, it also contains relations which would usually fall into the miscellaneous domain, e.g., award or publication. Also the data is derived from multiple sources, e.g., DBpedia and Freebase. (2) We aimed to not only cover relations for a fixed point in time (e.g., a person's birth date) but to also include relations which appear over a longer period of time (e.g., the marriage between persons). (3) We also tried to cover relations which appeared before the information age (before 1980). An overview of the most frequently occurring time points and time periods of the FactBench train set can be seen in Figure 8.

8.2.1. Positive Examples

In general, we use facts contained in DBpedia and Freebase as positive examples. Since for most relations there is far more data available then necessary we had to select a subset. For each of the properties we consider, we generated positive examples by issuing a SPARQL or MQL query and selecting the top 150 results. Note that the results in Freebase (MQL) are ordered by an internal relevance score[26]. The results for the DBpedia SPARQL queries were ordered by the number of inbound-links of a given resources' wikipedia page. We collected a total of 1500 correct statements (750 in test and train set). Each relation has 150 correct facts distributed equally in the test and train set.

8.2.2. Negative Examples

The generation of negative examples is more involved than the generation of positive examples. In order to effectively train any fact validation algorithm, we considered it essential that many of the negative examples are similar to true statements. In particular, most statements should be meaningful triples. For this reason, we derive negative examples from positive examples by modifying them while still following domain and range restrictions. Assume the input triple $t = (s, p, o)$ and the corresponding time period $tp = (from, to)$ in a knowledge base K is given and let S be the set of all subjects, O the set of all objects of the given property p and P the set of all properties. We used the following methods to generate the negative example sets dubbed subject, object, subject-object, property, random, mix and date (in that order). If not stated otherwise tp is not modified.

domain A triple (s', p, o) is generated where s' is a random element of S, the triple (s', p, o) is not contained in K.

range A triple (s, p, o') is generated analogously by selecting a random element of O.

domainrange A triple (s', p, o') is generated analogously by selecting a random s' from S and a random o' from O.

[26]http://wiki.freebase.com/wiki/Search_Cookbook

property A triple (s, p', o) is generated in which p' is randomly selected from the list of all properties and (s, p', o) is not contained in K and $p = p'$ is not allowed.

random A triple (s', p', o') is generated where s' and o' are randomly selected resources, p' is a randomly selected property from the list of all properties and (s', p', o') is not contained in K.

date A triple $(s, p, o)(from', to')$ is generated. For time points $from'$ is a random year drawn from a gaussian distribution ($\mu = from$ and $\sigma^2 = 5$), $from' = to'$, $from' \neq from$ and $0 < from' \leq 2013$. For timespans $from'$ is a random year drawn from a gaussian distribution ($\mu = from$ and $\sigma^2 = 2$), the duration d' is generated by drawing a random number from a gaussian distribution ($\mu = to - from$ and $\sigma^2 = 5$), $to' = from' + d'$, $0 < d' \leq 2013$, $from \neq from'$, $to \neq to'$, $from \leq 2013$ and $to \leq 2013$.

mix 1/6 of each of the above created negative training sets were randomly selected to create a heterogenous test set. Note that this set contains 780 negative examples.

9. Evaluation

The aim of our evaluation was three-fold. We wanted to quantify how well/much *a*) DeFacto can distinguish between correct and wrong facts; *b*) DeFacto is able to find correct time points or time periods for a given fact and if the year frequency distribution (1900 vs. 2013) does influence the accuracy; and *c*) the use of multi-lingual patterns boost the results of DeFacto with respect to fact validation and date detection. In the following, we describe how we set up our evaluation system, present the experiments we devised and discuss our findings.

9.1. Experimental Setup

In a first step, we computed all feature vectors, described in Section 6 for the training and test sets. DeFacto relies heavily on web requests, which are not deterministic (i.e., the same search engine query does not always return the same result). To achieve deterministic behavior and to increase the performance as well as reduce load on the servers, all web requests were cached. The DeFacto runtime for an input triple was on average slightly below four seconds per input triple[27] when using caches.

We stored the features in the ARFF file format and employed the WEKA machine learning toolkit[28] for training different classifiers. In particular, we were interested in classifiers which can handle numeric values and output

[27]The performance is roughly equal on server machines and notebooks, since the web requests dominate.

[28]http://www.cs.waikato.ac.nz/ml/weka/

```
@prefix fbase: <http://rdf.freebase.com/ns/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbo:   <http://dbpedia.org/ontology/> .
@prefix dbr:   <http://dbpedia.org/resource/> .
@prefix fr-dbr: <http://fr.dbpedia.org/resource/> .
@prefix de-dbr: <http://de.dbpedia.org/resource/> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .

fbase:m.0dt39
  rdfs:label    "Nobel Prize in Physics"@en, "Prix Nobel de physique"@fr, "Nobelpreis für Physik"@de ;
  owl:sameAs    fr-dbr:Prix_Nobel_de_physique , de-dbr:Nobelpreis_für_Physik , dbr:
      Nobel_Prize_in_Physics ;
  skos:altLabel "Nobel Physics Prize"@en , "Nobel laureates in physics"@fr , "Physik-Nobelpreis"@de
      ...

fbase:m.0jcx__24
  dbo:award  fbase:m.0dt39 ;
  dbo:startYear   "1921"^^xsd:gYear ;
  dbo:endYear     "1921"^^xsd:gYear .

fbase:m.0jcx
  rdfs:label          "Albert Einstein"@fr , "Albert Einstein"@en , "Albert Einstein"@de ;
  dbo:recievedAward   fbase:m.0jcx__24 ;
  owl:sameAs          dbr:Albert_Einstein , dbr-fr:Albert_Einstein , dbr-de:Albert_Einstein ;
  skos:altLabel       "A. Einstein"@fr , "Einstein, Albert"@de , "Albert Einstin"@en ...
```

Listing 1: Example of a fact in FactBench.

confidence values. Naturally, confidence values for facts such as, e.g., 95%, are more useful for end users than just a binary response on whether DeFacto considers the input triple to be true, since they allow a more fine-grained assessment. We selected popular machine-learning algorithms satisfying those requirements.

As mentioned in Section 4.1, we focused our experiments on the 10 relations from FactBench. The system can be extended easily to cover more properties by extending the training set of BOA to those properties. Note that DeFacto itself is also not limited to DBpedia or Freebase, i.e., while all of its components are trained on these datasets, the algorithms can be applied to arbitrary URIs and knowledge bases.

9.2. Fact Scoring

For this evaluation task, we used each FactBench training set to build an independent classifier. We then used the classifier on the corresponding test set to evaluate the built model on unseen data. The results on this task can be seen in Table 5. The J48 algorithm, an implementation of the C4.5 decision tree – shows the most promising results. Given the challenging tasks, F-measures up to 84.9% for the *mix* test set appear to be very positive indicators that DeFacto can be used to effectively distinguish between true and false statements, which was our primary evaluation objective. In general, DeFacto also appears to be stable against the various negative test sets given the F_1 values ranging from 89.7% to 91% for the *domain*, *range*, *domainrange* and *random* test set. In particular, the algorithms with overall positive results also seem less affected by the different variations. On the *property* test set, in our opinion the hardest task, we achieved an F_1 score of 68.7%. Due to the results achieved, we use J48 as the main

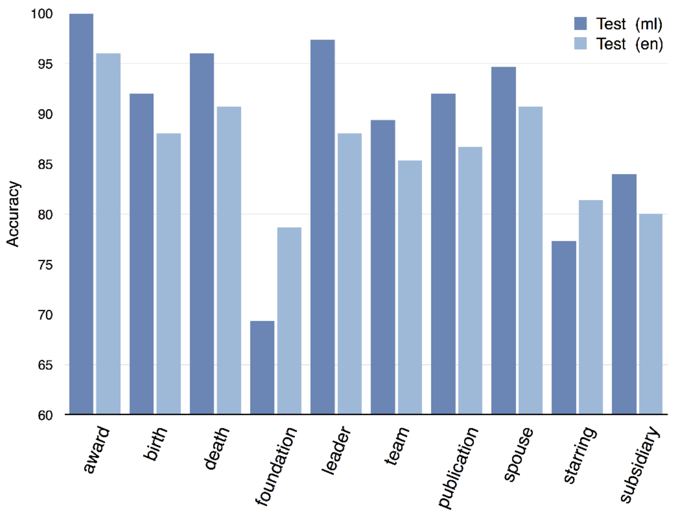


Figure 9: Accuracy results for learned J48 *mix* classifier on correct subset of the test set. The abbreviation *ml* indicates that multi-lingual (English, French, German) search results and surface forms were used, *en* is limited to English only.

classifier in DeFacto and, more specifically, its results on the mix sets as this covers a wide range of scenarios. We observe that the learned classifier has an error rate of 3% for correct facts, but fails to classify 55.3% of the false test instances as incorrect.

We also performed an evaluation to measure the performance of the classifier for each of the relations in FactBench. The results of the evaluation are shown in Figure 9. We used the precision of the main classifier (J48 on the *mix* models) on the correct subset for this figure.[29] The average precision for all relations is 89.2%. The worst

[29]We are using the correct subset, since some negative examples

Relation	|Sub|	|Obj|	Type	Year_{min}	Year_{max}	Year_{avg}	Source	Comment
birth	75/75	67/65	point	1166/1650	1989/1987	1925/1935	DBpedia	birth place (city) and date of persons
death	75/75	54/48	point	1270/1677	2013/2012	1944/1952	DBpedia	death place (city) and date of persons
team	50/52	24/27	point	2001/2001	2012/2012	2007/2007	DBpedia	NBA players for a NBA team (after 2000)
award	75/75	5/5	point	1901/1901	2007/2007	1946/1952	Freebase	winners of nobel prizes
foundation	75/75	59/62	point	1865/1935	2006/2008	1988/1990	Freebase	foundation place and time of software companies
publication	75/75	75/73	point	1818/1918	2006/2006	1969/1980	Freebase	authors of science fiction books (one book/author)
spouse	74/74	74/74	point	2003/2003	2013/2013	2007/2007	Freebase	marriages between actors (after 2013/01/01)
starring	22/21	74/74	period	1954/1964	2009/2009	1992/1993	DBpedia	actors starring in a movie
leader	75/75	36/43	period	1840/1815	2013/2012	1973/1972	DBpedia	prime ministers of countries
subsidiary	54/50	75/75	period	1993/1969	2007/2007	2003/2002	Freebase	company acquisitions

Table 4: Overview of all correct facts of the training and testing set (train/test).

	Domain						Range					
	C	P	R	F_1	AUC	RMSE	C	P	R	F_1	AUC	RMSE
J48	89.7%	0.898	0.897	0.897	0.904	0.295	90.9%	0.909	0.909	0.909	0.954	0.271
SimpleLogistic	89.0%	0.890	0.890	0.890	0.949	0.298	88.0%	0.880	0.880	0.880	0.946	0.301
NaiveBayes	81.2%	0.837	0.812	0.808	0.930	0.415	83.3%	0.852	0.833	0.830	0.933	0.387
SMO	85.4%	0.861	0.854	0.853	0.854	0.382	83.3%	0.852	0.833	0.830	0.833	0.409
	DomainRange						Property					
	C	P	R	F_1	AUC	RMSE	C	P	R	F_1	AUC	RMSE
J48	91.0%	0.910	0.910	0.910	0.953	0.270	70.8%	0.786	0.708	0.687	0.742	0.427
SimpleLogistic	88.9%	0.889	0.889	0.889	0.950	0.296	64.9%	0.653	0.649	0.646	0.726	0.460
NaiveBayes	84.5%	0.861	0.845	0.843	0.935	0.380	61.3%	0.620	0.613	0.608	0.698	0.488
SMO	83.6%	0.853	0.836	0.834	0.836	0.405	64.6%	0.673	0.646	0.632	0.646	0.595
	Random						Mix					
	C	P	R	F_1	AUC	RMSE	C	P	R	F_1	AUC	RMSE
J48	90.9%	0.910	0.909	0.909	0.933	0.283	84.9%	0.850	0.849	0.849	0.868	0.358
SimpleLogistic	87.8%	0.879	0.878	0.878	0.954	0.293	80.2%	0.810	0.802	0.799	0.880	0.371
NaiveBayes	84.1%	0.851	0.841	0.839	0.942	0.375	78.7%	0.789	0.787	0.787	0.867	0.411
SMO	84.3%	0.864	0.843	0.841	0.843	0.396	76.9%	0.817	0.769	0.756	0.754	0.480

Table 5: Classification results for FactBench test sets (C = correctness, P = precision, R = recall, F_1 = F_1 Score, AUC = area under the curve, RMSE = root mean squared error).

precision for an individual relation, i.e., 69%, is achieved on the *foundation* relation, which is by far the least frequent relation on the Web with respect to search engine results.

9.3. Date Scoring

To estimate time scopes, we first needed to determine appropriate parameters for this challenging task. To this end, we varied the context size from 25, 50, 100 and 150 characters to the left and right of the proofs subject and object occurrence. Additionally, we also varied the used languages which is discussed in more detail in Section 9.4. The final parameter in this evaluation was the normalization approach. As introduced in Section 7, we used the occurrence (number of occurrences of years in the context for all proofs of a fact), the domain and range approach. We performed a grid search for the given parameters on the *correct* train set. As performance measures we choose

are generated by replacing properties as described in Section 8.2.2. For those, it would not be clear, which property they refer to.

precision[30] P (shown in Equation 14), recall R (shown in Equation 15) and F-measure, defined as $F_1 = 2 * \frac{P*R}{P+R}$.

$$P = \frac{|relevant\ years \cap retrieved\ years|}{|retrieved\ years|} \quad (14)$$

$$R = \frac{|relevant\ years \cap retrieved\ years|}{|relevant\ years|} \quad (15)$$

If for example, for a single fact the correct time period is 2008 (a time point), the F_1 score is either 0 or 1. However, if the correct time period is $2011-2013$ and the retrieved results are $2010-2013$, we would achieve a precision $P = \frac{3}{4}$ (three of the four retrieved years are correct) and a recall $R = 1$ (all of the relevant years were found), resulting in an F_1 score of $\frac{6}{7}$.

The final results for the train set are shown in Table 6. Please note that it is out of scope of this paper to decide whether a given property requires a time period or a time point. As expected, facts with time point show a higher F_1 measure as facts with time period. Calculating the average F_1 score for the individual relations leads to $F_1 = 70.2\%$ for time points and $F_1 = 65.8\% F_1$ for relations associated with time periods. The relations performing well on fact scoring also appear to be better suited for year scoping, e.g., the *award* relation. In general, the training results show that the *domain* normalization performs best and the optimal context size varies for each relation. We now applied the learned parameters for each relation on the FactBench *correct* test subset. The results are shown in Table 7. The average F_1 score decreases by 2.5% to 67.7% for time points and 4.6% to 61.2% for time period relations compared to the train set. Since it is not possible to determine a correct time point or time period for all facts (the context does not always include the correct year(s)) we also calculated DeFacto's accuracy. We define the accuracy *acc* for a time period *tp* as follows:

$$acc(tp) = \begin{cases} 1 & \text{if } tp_{from}\ is\ correct \wedge tp_{to}\ is\ correct \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

The average accuracy for time point (from and to are equal) relations is 76%. Since for time periods we have to match both start and end year, which aggravates this task significantly, we achieved an accuracy of 44% on this dataset. Finally, we wanted to see if DeFacto's performance is influenced by how recent a fact is. We grouped the time intervals in buckets of 10 years and plotted the proportion of correctly classified facts within this interval. We did this for the multilingual as well as the English-only setting of DeFacto. The results are shown in Figure 10. In general, all values are between 80% and 100% for the English version and between 93% and 100% for the mulitlingual version. While there is some variation, no obvious correlation can be observed, i.e., DeFacto appears to be able to handle recent and older facts. In this figure, it is interesting to note that the multilingual setting appears to be more stable and perform better. We performed a paired t-test using all 750 facts and obtained that the improvement of the multilingual setting is statistically very significant.

[30] Finding no year candidate for a given fact only influences the recall.

$\text{Set}^{context}_{language}$	P	R	F	MRR	C_S	C_E	P_{75}	Acc
award^{100}_{en}	93.3	93.3	93.3	100	70	-	75	93.3
award^{25}_{ml}	93.3	93.3	93.3	100	70	-	75	93.3
birth^{50}_{en}	77.8	74.7	76.2	81.6	56	-	69	81.2
birth^{25}_{ml}	93.2	92	92.6	93.3	69	-	73	94.5
death^{25}_{en}	72	72	72	84.5	54	-	69	78.3
death^{25}_{ml}	81.3	81.3	81.3	87.1	61	-	74	82.4
$\text{foundation}^{150}_{en}$	22.2	18.7	20.3	66.1	14	-	20	70
$\text{foundation}^{150}_{ml}$	20.3	18.7	19.4	48.1	14	-	33	42.4
$\text{publication}^{150}_{en}$	62	58.7	60.3	77.8	44	-	68	64.7
$\text{publication}^{150}_{ml}$	67.6	66.7	67.1	75.5	50	-	74	67.6
$\text{starring}^{50}_{en}$	57.1	48	52.2	87.1	36	-	44	81.8
$\text{starring}^{100}_{ml}$	61.4	57.3	59.3	73.6	43	-	60	71.7
$\text{subsidiary}^{150}_{en}$	60.7	49.3	54.4	79.3	37	-	53	69.8
$\text{subsidiary}^{25}_{ml}$	70.2	53.3	60.6	87.5	40	-	50	80
spouse^{25}_{en}	69.2	59	63.6	-	34	35	34	76.5
spouse^{25}_{ml}	73.7	61.4	67	-	36	36	42	59.5
team^{150}_{en}	52.7	42.7	47.2	-	25	16	51	23.5
team^{25}_{ml}	59.9	49.6	54.3	-	28	16	45	26.7
leader^{100}_{en}	46.3	60.8	52.5	-	29	29	56	44.6
leader^{100}_{ml}	55	71.5	62.2	-	38	37	72	45.8
$\text{timepoint}^{25}_{en}$	62	52	56.6	85.8	273	273	356	76.7
$\text{timepoint}^{25}_{ml}$	66.9	60.6	63.6	87.1	318	318	404	78.7
$\text{timeperiod}^{100}_{en}$	55.7	55.6	55.7	-	92	82	159	41.5
$\text{timeperiod}^{100}_{ml}$	59.6	60.1	59.8	-	102	91	195	38.5
all^{100}_{en}	58.2	54.4	56.2	-	375	365	563	62
all^{100}_{ml}	61.5	59.6	60.5	-	414	403	634	61

Table 7: Overview of the *domain*-normalization on the FactBench test set. *ml* (multi-lingual) indicates the use of all three languages (en,de,fr). $C_{(S|E)}$ shows the number of correct start and end years, P_{75} is the number of time-periods possible to detect correctly and A is the accuracy on P_{75}.

9.4. Effect of Multi-lingual Patterns

The last question we wanted to answer in this evaluation is how much the use of the multi-lingual patterns boosts the evidence scoring as well as the date scoping.

	occurrence						global						domain					
Set	C	P	R	F	P_{75}	A	C	P	R	F	P_{75}	A	C	P	R	F	P_{75}	A
award$_{en}$	25	100	98.7	**99.3**	74	100	25	98.6	97.3	**98**	74	98.6	100	100	98.7	**99.3**	74	100
award$_{ml}$	25	100	98.7	**99.3**	74	100	25	100	98.7	**99.3**	74	100	25	100	98.7	**99.3**	74	100
birth$_{en}$	25	83.3	80	81.6	69	87	50	91.7	88	**89.8**	70	94.3	50	76.4	73.3	74.8	70	78.6
birth$_{ml}$	50	93.2	92	92.6	73	94.5	25	94.6	93.3	**94**	73	95.9	25	89.2	88	88.6	73	90.4
death$_{en}$	50	74.3	73.3	73.8	69	79.7	25	61.1	58.7	59.9	68	64.7	25	80.6	77.3	**78.9**	68	85.3
death$_{ml}$	25	77.3	77.3	77.3	75	77.3	25	66.7	66.7	66.7	75	66.7	25	84	84	**84**	75	84
foundation$_{en}$	150	14.1	12	12.9	28	32.1	150	17.2	14.7	15.8	28	39.3	150	25	21.3	**23**	28	57.1
foundation$_{ml}$	25	16.4	13.3	14.7	23	43.5	150	21.7	20	20.8	41	36.6	150	26.1	24	**25**	41	43.9
publication$_{en}$	100	58.3	56	57.1	63	66.7	150	60.3	58.7	**59.5**	67	65.7	100	51.4	49.3	50.3	63	58.7
publication$_{ml}$	25	70.8	68	69.4	68	75	150	74.7	74.7	**74.7**	72	77.8	50	60	60	60	70	64.3
starring$_{en}$	25	64.4	38.7	48.3	35	82.9	50	67.9	48	**56.3**	40	90	100	59.3	46.7	52.2	46	76.1
starring$_{ml}$	25	59.6	45.3	51.5	44	77.3	50	58.1	48	52.6	48	75	100	62.7	56	**59.2**	57	73.7
subsidiary$_{en}$	100	63.5	44	52	45	73.3	50	63	38.7	47.9	39	74.4	150	64.8	46.7	**54.3**	46	76.1
subsidiary$_{ml}$	25	70.8	45.3	**55.3**	43	79.1	25	68.8	44	53.7	43	76.7	25	70.8	45.3	**55.3**	43	79.1
spouse$_{en}$	100	67.5	68	67.7	53	50.9	25	75.5	64.4	69.5	37	78.4	25	77.1	65.2	**70.6**	37	78.4
spouse$_{ml}$	25	69.6	66.5	68	49	59.2	25	70.8	65.6	68.1	49	55.1	25	75.2	67.2	**71**	49	61.2
nbateam$_{en}$	100	54.2	47.4	50.6	44	34.1	100	57.8	47	51.9	44	34.1	150	59.1	48.4	**53.2**	53	28.3
nbateam$_{ml}$	50	60.2	58.1	59.1	58	25.9	100	62.1	55.4	58.6	63	23.8	25	65.2	58.7	**61.8**	53	32.1
leader$_{en}$	100	42.6	65.1	51.5	55	41.8	100	42.6	63.1	50.9	55	41.8	100	46.7	64.4	**54.1**	55	43.6
leader$_{ml}$	100	53.6	75.4	62.6	72	44.4	100	53.3	75.6	62.5	72	44.4	100	55.9	76.7	**64.7**	72	45.8
timepoint$_{en}$	25	61	48	**53.7**	277	78	25	60.2	47.3	53	277	76.9	100	57.8	50	53.6	317	71
timepoint$_{ml}$	25	65.9	56.7	**60.9**	326	78.2	25	64.1	55.1	59.3	326	76.1	150	61.6	58.2	59.9	373	70.2
timeperiod$_{en}$	100	54.7	60.2	57.3	152	42.8	100	54.9	60.3	57.4	152	42.8	100	58.7	60.6	**59.7**	152	44.7
timeperiod$_{ml}$	100	59	67.2	62.8	198	38.9	100	59.4	67.5	63.2	198	39.4	100	63	69	**65.9**	198	40.9
all$_{en}$	50	61.3	56.2	58.6	496	72	25	64	54	58.6	460	75.4	100	62.7	58.1	**60.3**	543	67.6
all$_{ml}$	25	67.1	63.2	65.1	568	70.1	25	66.3	62.4	64.3	568	68.7	100	66.1	65.2	**65.7**	635	65.4

Table 6: Overview of the time-period detection task for the FactBench training set with respect to the different normalization methods. *ml* (multi-lingual) indicates the use of all three languages (en,de,fr).

	C	P	R	F_1	AUC	RMSE
J48	83.4%	0.834	0.834	**0.834**	0.877	0.361
SimpleLogistic	80.6%	0.811	0.806	0.804	0.884	0.368
NaiveBayes	78.1%	0.788	0.781	0.782	0.872	0.428
SMO	78.6%	0.816	0.786	0.777	0.773	0.463

Table 8: Classification results for FactBench mix test set on English language only.

For the fact scoring we trained different classifiers on the *mix* training set. We only used English patterns and surface forms to extract the feature vectors. As the results in Table 8 on the test set show, J48 is again the highest scoring classifier, but is outperformed by the multi-lingual version shown in Table 5 by 1.5% F_1 score.

The detailed analysis for the different relations in Figure 9 indicates a superiority of the multi-lingual approach.

We also performed the grid search as presented in Section 9.3 for English patterns and surface forms only. As shown in Table 6 the multi-lingual date scoping approach outperforms the English one significantly on the training set. The multi-lingual version achieved an average 4.3% on the time point and a 6.5% better F_1 measure on time period relations.

The difference is similar on the test set, where the difference is 6.5% for time points and 6.9% for time period relations.

Finally, as shown in Figure 10, the English version performs equally well on recent data, but performs worse for less recent dates, which is another indicator that the use of a multilingual approach is preferable to an English-only setting.

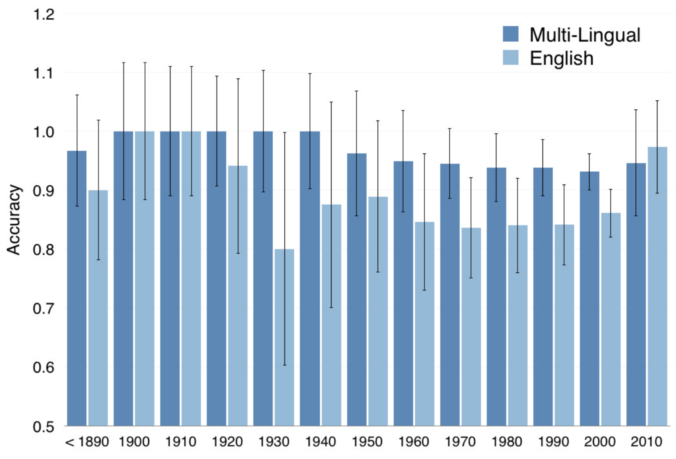


Figure 10: A plot showing the proportion of correctly classified facts (y-axis) for the FactBench *mix-correct*-test-set using the J48 classifier. The time intervals (x-axis) are buckets of ten years, e.g., 1910 stands for all years from 1910 to 1919. Results for the multilinguael and English-only setting of DeFacto are shown.

10. Conclusion and Future Work

In this paper, we presented DeFacto, a multilingual and temporal approach for checking the validity of RDF triples using the Web as corpus. In more detail, we explicated how multi-lingual natural-language patterns for formal relations can be used for fact validation. In addition, we presented an extension for detecting the temporal scope of RDF triples with the help of pattern and frequency analysis. We support the endeavour of creating better fact validation algorithms (and to that end also better relation extraction and named entity disambiguation systems) by providing the full-fledged benchmark FactBench. This benchmarks consists of one training and several test sets for fact validation as well as temporal scope detection. We showed that our approach achieves an F_1 measure of 84.9% on the most realistic fact validation test set (FactBench *mix*) on DBpedia as well as Freebase data. The temporal extension shows a promising average F_1 measure of 70.2% for time point and 65.8% for time period relations. The use of multi-lingual patterns increased the fact validation F_1 by 1.5%. Moreover, it raised the F_1 for the date scoping task of up to 6.9%. Of importance is also that our approach is now fit to be used on non-English knowledge bases.

Our approach can be extended in manifold ways. First, we could run the experiments on a Web crawl such as ClueWeb09[31]/ClueWeb12[32] or CommonCrawl[33].

This would drastically increase recall, since we could execute all combinations of subject/object surface forms and patterns as well as precision, since we could also query for exact matches like ``Albert Einstein was awarded the Nobel Prize in Physics'' as opposed to querying for fragments (see Section 4.2). Second, we could work on efficient disambiguation of (parts of) the web page's text before extracting proof phrases. This would be useful for, e.g., differentiating between Winston Churchill, the American novelist and Winston Churchill the British prime minister. Third, we plan to analyse the influence of individual features, as well as our threshold optimization and further relations among the model components. Besides, check individually the coverage of NLP tools. Moreover, we plan to extend the approach to combine more languages which were not considered at this work, such as Spanish and Portuguese, for instance. This would increase the coverage of possible facts in many cases and consequently improving the results. Furthermore, we could extend our approach to support data type properties or try to search for negative evidence for facts, therewith allowing users to have a richer view of the data on the Web through DeFacto. Finally, we could extend the user interface (see Figure 4) to improve classifier performance by incorporating a feedback loop allowing users to vote on overall results, as well as proofs found on web pages. This feedback can then be fed into our overall machine learning pipeline and improve DeFacto on subsequent runs.

Acknowledgments. This work was supported by grants from the European Union's 7th Framework Programme provided for the projects GeoKnow (GA no. 318159), the Eurostars project DIESEL as well as the German Research Foundation Project GOLD and the German Ministry of Economy and Energy project SAKE (GA No. 01MD15006E).

References

Agichtein, E., Gravano, L., 2000. Snowball: Extracting relations from large plain-text collections, in: In ACM DL, pp. 85–94.

Augenstein, I., Padó, S., Rudolph, S., 2012. Lodifier: Generating linked data from unstructured text., in: ESWC, pp. 210–224.

Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J., 2012. PROV-O: The PROV Ontology. Technical Report. URL: `http://www.w3.org/TR/prov-o/`.

Brin, S., 1999. Extracting patterns and relations from the world wide web, in: WebDB, pp. 172–183.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E.R.H., Mitchell, T.M., 2010. Toward an architecture for never-ending language learning, in: Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010).

Correndo, G., Salvadores, M., Millard, I., Shadbolt, N., 2010. Linked timelines: Temporal representation and management in linked data., in: COLD.

Dividino, R., Sizov, S., Staab, S., Schueler, B., 2011. Querying for Provenance, Trust, Uncertainty and other Meta Knowledge in RDF. Web Semantics: Science, Services and Agents on the World Wide Web 7. URL: `http://www.websemanticsjournal.org/index.php/ps/article/view/168`.

Dong, X.L., Berti-Equille, L., Srivastava, D., 2009. Truth discovery and copying detection in a dynamic world. PVLDB 2, 562–573.

Galland, A., Abiteboul, S., Marian, A., Senellart, P., 2010. Corroborating information from disagreeing views., in: WSDM, ACM. pp. 131–140.

Gerber, D., Ngonga Ngomo, A.C., 2011. Bootstrapping the linked data web, in: 1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011.

[31]`http://lemurproject.org/clueweb09`
[32]`http://lemurproject.org/clueweb12`
[33]`http://commoncrawl.org/`

Gerber, D., Ngonga Ngomo, A.C., 2012. Extracting Multilingual Natural-Language Patterns for RDF Predicates, in: Proceedings of EKAW.

Grishman, R., Yangarber, R., 1998. Nyu: Description of the Proteus/Pet system as used for MUC-7 ST, in: MUC-7, Morgan Kaufmann.

Gutierrez, C., Hurtado, C., Vaisman, A., 2005. Temporal rdf, in: The Semantic Web: Research and Applications. Springer, pp. 93–107.

Hartig, O., 2008. Trustworthiness of data on the web, in: Proceedings of the STI Berlin & CSW PhD Workshop.

Hartig, O., 2009. Provenance information in the web of data, in: Proceedings of LDOW.

Hartig, O., Zhao, J., 2010. Publishing and consuming provenance metadata on the web of linked data, in: IPAW, pp. 78–90.

Hellmann, S., Lehmann, J., Auer, S., Brümmer, M., 2013. Integrating NLP using Linked Data, in: Submitted to 12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia.

Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G., 2011. Robust Disambiguation of Named Entities in Text, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 782–792.

Jain, A., Pantel, P., 2010. Factrank: Random walks on a web of facts, in: In Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010, pp. 501–509. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.230.8852`.

Kleinberg, J.M., 1999. Hubs, authorities, and communities. ACM Comput. Surv. .

Krause, S., Li, H., Uszkoreit, H., Xu, F., 2012. Large-scale learning of relation-extraction rules with distant supervision from the web., in: International Semantic Web Conference, pp. 263–278.

Landwehr, N., Hall, M., Frank, E., 2005. Logistic model trees 95, 161–205.

Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S., 2009. DBpedia - a crystallization point for the web of data. Journal of Web Semantics 7, 154–165.

Lehmann, J., Gerber, D., Morsey, M., Ngonga Ngomo, A.C., 2012. Defacto - deep fact validation, in: Proc. of the International Semantic Web Conference. URL: `http://jens-lehmann.org/files/2012/iswc_defacto.pdf`.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C., 2014. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal .

Meiser, T., Dylla, M., Theobald, M., 2011. Interactive reasoning in uncertain RDF knowledge bases, in: Berendt, B., de Vries, A., Fan, W., Macdonald, C. (Eds.), CIKM'11, pp. 2557–2560.

Mendes, P.N., Jakob, M., Garcia-Silva, A., Bizer, C., 2011. DBpedia Spotlight: Shedding Light on the Web of Documents, in: Proceedings of I-SEMANTICS.

Nakamura, S., Konishi, S., Jatowt, A., Ohshima, H., Kondo, H., Tezuka, T., Oyama, S., Tanaka, K., 2007. in: ECDL, pp. 38–49.

Nakashole, N., Theobald, M., Weikum, G., 2011. Scalable knowledge harvesting with high precision and high recall., ACM. pp. 227–236.

Navigli, R., Ponzetto, S.P., 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif. Intell. , 217–250.

Nguyen, D.P.T., Matsuo, Y., Ishizuka, M., 2007. Relation extraction from wikipedia using subtree mining, in: AAAI, pp. 1414–1420.

Pasternack, J., Roth, D., 2011a. Generalized fact-finding., in: Proceedings of WWW (Companion Volume), ACM. pp. 99–100.

Pasternack, J., Roth, D., 2011b. Making better informed trust decisions with generalized fact-finding., in: Proceedings of IJCAI, pp. 2324–2329.

Pasternack, J., Roth, D., 2013. Latent credibility analysis, in: Proceedings of the 22Nd International Conference on World Wide Web, pp. 1009–1020.

Rula, A., Palmonari, M., Harth, A., Stadtmüller, S., Maurino, A., 2012. On the diversity and availability of temporal information in linked open data, in: The 11th International Semantic Web Conference (ISWC2012).

Talukdar, P.P., Wijaya, D., Mitchell, T., 2012a. Acquiring temporal constraints between relations, in: Proceedings of the Conference on Information and Knowledge Management (CIKM 2012), Association for Computing Machinery, Hawaii, USA.

Talukdar, P.P., Wijaya, D., Mitchell, T., 2012b. Coupled temporal scoping of relational facts, in: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM), Association for Computing Machinery, Seattle, Washington, USA.

Theoharis, Y., Fundulaki, I., Karvounarakis, G., Christophides, V., 2011. On Provenance of Queries on Semantic Web Data. IEEE Internet Computing 15, 31–39.

Usbeck, R., Ngomo, A.C.N., Röder, M., Gerber, D., Coelho, S., Auer, S., Both, A., 2014. AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data, in: The Semantic Web – ISWC 2014. Springer International Publishing. volume 8796 of *Lecture Notes in Computer Science*, pp. 457–471. doi:`10.1007/978-3-319-11964-9_29`.

Wang, Y., Yang, B., Qu, L., Spaniol, M., Weikum, G., 2011. Harvesting Facts from Textual Web Sources by Constrained Label Propagation, in: Proceedings of the 20^{th} ACM Conference on Information and Knowledge Management (CIKM), Glasgow, Scotland, UK, October 24-28, 2011, pp. 837–846.

Wang, Y., Zhu, M., Qu, L., Spaniol, M., Weikum, G., 2010. Timely YAGO: harvesting, querying, and visualizing temporal knowledge from Wikipedia., in: EDBT, ACM. pp. 697–700.

Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., Ishizuka, M., 2009. Unsupervised relation extraction by mining wikipedia texts using information from the web, in: ACL, pp. 1021–1029.

Yin, X., Han, J., Yu, P.S., 2007. Truth discovery with multiple conflicting information providers on the web, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1048–1052.

Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S., Hitzler, P., 2015. Quality assessment methodologies for linked open data. Semantic Web Journal .