FactCheck - Identify and Fix Conflicting Data on the Web

Peter Kalchgruber(✉), Wolfgang Klas, Nour Jnoub, and Elaheh Momeni

Faculty of Computer Science, University of Vienna, Vienna, Austria
{peter.kalchgruber,wolfgang.klas,nour.jnoub,elaheh.momeni}@univie.ac.at

Abstract. Today's Web pages more frequently contain structured information by means of semantically-rich embedded data. These data are currently used by search engines to provide an improved semantic, structured search result. However, very often these data are conflicting when present on different Web pages. Data consumers do not receive any support in handling (recognizing, interpreting, correcting) structured data. The more frequent structured data will be available on the Web, the more critical is the problem in recognizing and handling conflicting structured data. This paper presents *FactCheck* as an approach for the detection and resolution of conflicting structured data on the Web. We applied our solution to websites and APIs using JSON-LD and Microdata. First experiments with a prototypical implementation of *FactCheck* show that the resolution of conflicting data has high potential to significantly improve the quality of information on the Web. First evaluation reveals positive effects through the additional information indicators provided by *FactCheck* and indicates an impact on users of *FactCheck* at various levels. The resulting improvement of the data quality on websites will be of benefit to all data consumers who are depending on data on the Web as well as cognitive computing services and its building blocks.

Keywords: Web infrastructure · Conflicting web data
Structured data · Semantic web

1 Introduction

When looking at structured data, i.e., semantically-rich embedded data on the Web, it is likely to observe that publishing structured data was simplified a lot by the introduction of various semantic markup capabilities (e.g., Microdata [9], RDFa [1], Microformats[1], JSON-LD [15]). The development of structured data on the Web was further advanced by the introduction of schema.org [6]. The fact that leading search engines [5,7] widely accept data markup advances the development of structured data on the Web. As a result, the amount of structured

[1] http://microformats.org/.

T. Mikkonen et al. (Eds.): ICWE 2018, LNCS 10845, pp. 312–320, 2018.
https://doi.org/10.1007/978-3-319-91662-0_25

data is increasing drastically. Popular examples using structured data markups can be found in a wide variety of well-known websites[2].

Growth rates of structured data around factor four per year are no exception [12]. We expect the amount of structured data to increase further and expect structured data to constitute an essential part of every website in the future. Because of the already large amount of structured data available, the number of data consumers depending on such structured data will also increase.

However, despite the wide-spread adoption and the growing prevalence of data consumers, structured data in the Web faces a significant problem: In the course of our work it turned out that the quality of structured data is quite challenging in terms of consistency and validity. The use and trust in applications and services (e.g., knowledge graphs) built upon that data is therefore partly to be called in question. If, for instance, a content provider publishes outdated or false facts, content consumers who rely on structured data may draw false conclusions. This happened during the Austrian presidential election campaign, when Google Search falsely identified the loser of the second ballot as the new Austrian president[3]. Data fusion addresses this problem.

To solve this problem, we propose a framework with unique features which enables the user to: (i) become aware of conflicting structured data so that she can recognize a potential problem; (ii) confirm correctness or defectiveness of structured data; (iii) give feedback to data owners, analytic software, Web services, or crawler, in order to confirm correctness or to provide accurate data for the rectification of defective structured data; (iv) be assisted during the creation and updating process of contents in content management systems with structured data assistance [11] in order to prevent that contents are associated with conflicting structured data.

2 Related Work

According to [3], data fusion is defined as the process of cleaning, aggregating and integrating data of an entity available from various data providers into a single clean consistent representation. For example, the Linked Data Integration Framework (LDIF) [14] applies data fusion for data in the format of RDF. Based on smart algorithms LDIF runs through all phases of data fusion described above and generates a clean consistent representation of RDF resources in form of an N-Quad file, optionally with provenance information and quality scores for all graphs. To some extend, approaches of data fusion aim to assist the user not to be bothered any more by conflicting values of the same entities. However, in the strict sense, these approaches create an n+1st representation of a named entity

[2] Some example domains for products, events, movies, music, videos and news: ebay.com, guardian.com, imdb.com, last.fm, nytimes.com, rottentomatoes.com, yelp.com, youtube.com ....

[3] Google falsely announced Norbert Hofer as the new president of Austria. The mistake was fixed on the next day by removing the answer box. However, it took almost one year to show the new elected president of Austria in the answer box.

while not fixing the conflicting data in the original source, where the data are consumed.

Another example of online data fusion is the Google Knowledge Vault. Google continuously performs online data fusion with structured data of the Web and updates its Knowledge Vault based on smart algorithms [4,13]. As seen in the example above, during the Austrian presidential elections, these algorithms fail sometimes. Again, this approach does not address the problem of fixing the conflicting source data. Other prominent examples are the user-driven databases Wikidata [18] and DBpedia [2]. In Linked Open Data, these data sets are often seen as an "enrichment" hub for many websites. Although they are adequately connected with other data providers which are offering information about the same (real-world) entities, the exchange of information works only partially and often only in one direction (e.g., from Wikipedia to DBpedia).

Hence, potentially isolated data spheres may result from these deficiencies. For example, Wikipedia only tends to react to changes in equivalent resources of other data providers because changes are incorporated manually by human editors.

Most importantly, in all these approaches, data fusion does not exploit the auspicious potential of the following three factors: resolving inconsistencies by using Linked Data; opportunities of a crowd-based approach to identify and fix inconsistencies; and sharing knowledge about the resolution of inconsistencies. These factors would help to fix conflicting data within the source.

3 Overview of the FactCheck Framework

3.1 Definitions

Let us assume that, for example, on the Web page of IMDb[4] a user is visiting information about a real-world entity, say the movie Flubber encoded in HTML, structured data associated and encoded in Microdata format. We define the Web page http://example.com/flubber as an URL of the data source s (e.g., IMDb, RottenTomatoes[5], New York Times[6]) and the movie Flubber as representation of the named entity as object o. Let us assume this data source has associated structured data described by means of markup of the representation object o. The structured data are composed of facts f, for example a published date which can carry some value (e.g., of type DateTime) of the set of valid values $V_{PublishedDate}$.

Very often data sources are specialized in a specific subject area (e.g., websites with information about movies, IMDb, RottenTomatoes). But there also exist websites covering a very broad spectrum of subject areas (e.g., Wikipedia, New York Times). In general, each data source can offer information about various objects. The function *objects* returns all objects that are provided by

[4] http://imdb.com.
[5] http://rottentomatoes.com.
[6] http://data.nytimes.com.

a data source. In order to obtain all facts of a given object the function *fact* returns all facts the given object possesses. The function *related* returns related objects of a given object (e.g., the representation of the (real-world) entity Flubber of IMDb data source is related to the representation of the same (real-world) entity at the data source of RottenTomatoes). The function *factValue* finally returns the value for a given pair consisting of an object and a fact (e.g., $factValue(\text{o} = \text{'Flubber@IMDb'}, \text{f} = \text{'ReleaseDate'}) \rightarrow \text{v} = \text{'1987-01-01'})$.

A function *relConf* basically investigates some fact/value-pairs of the objects to be compared or may use more richer contextual information about the objects, like solutions implemented in the course of current cognitive computing services (e.g., those used by the IBM Watson services, Microsoft Cognitive Computing Services, Google Services, ...) which seem to make use of algorithms that can reconciliate entities quite adequately (see also for example approaches described in [8,10,17]).

The detection of conflicting structured data, i.e., values of facts of various representations about same (real-world) entities, calls for a proper notion of similarity of the individual facts associated to objects. For one movie there could be manifold values, e.g., for the fact ReleaseDate: 1987-07-20, 1987-07-17, 20.07.1987, 1987-07-20T15:33+01:00.

We can not necessarily easily determine if the various values of the ReleaseDate above are identical, similar, or conflicting without a specific semantic context. The values vary in accuracy and formats but also in the true value. In terms of data quality the assessment of this variation of values is often referred to as "fitness for use" which implies that the user or use case determines the quality of the data [16,19]. We therefore define that **similarity** between two fact values of related objects always depends on the observer or use case the data will be needed. Therefore we classify facts by similarity predicates which determine the similarity of two values. For a better overview we group the predicates into different predicate categories: generic, abstraction, interval, date and language.

Each category consists of a well-defined set of predefined predicates (see Definition 1). E.g., the *generic* predicate category has a similarity predicate $p_{1.1}$ that can testify if two values are totally equal and identical (same format, same unit, same accuracy, same worth), or, e.g., the predicate $p_{1.2}$ could testify, if the base value is smaller than the comparison value.

Definition 1. *Let SPC denote the set of similarity predicate categories, then* c_i, $i = 1 \ldots SPC_{max}$, $c_i \in SPC$, $SPC_{max} = 5$, *is a predefined similarity predicate category, with* $c_i = \{p_{i.j} \mid p_{i.j}$ *is a predicate (mathematical logic),* $j \in \mathbb{N}\}$

Predicates of *FactCheck* allow for an automatic detection of conflicting values during data reconciliation. Predicates can determine if two given values (A, B) are conflicting or non-conflicting. E.g., a possible predicate $p_{4.3}$ of the predicate category c_4 of dates could claim: If B is in range of three days of A it is non-conflicting. Another predicate e.g., $p_{2.6}$ of the category of abstraction c_2 could claim: If, in a given taxonomy, B is a child of A the values are non-conflicting (e.g., A = "red", B = "bright red", with a given taxonomy that is about classifying colors).

In order to handle the potentially big variety of predicates over all facts we introduce the concept of the *FactCheck* precision metric (see Definition 2). A precision metric allows the smart combination of similarity predicates by logical operators. E.g. $p_{2.4} \wedge p_{4.7} \vee p_{2.5}$

Each fact f_i can be assigned a precision metric class pm (see Definition 3), e.g., the fact *ReleaseDate* can be assigned a precision metric class $pm_{ReleaseDate}$ based on a similarity predicate $p_{4.1}$ that e.g., determines that date values may only differ in a defined number of days in order not to constitute conflicting data.

Definition 2. *Domain PM denotes the set of possible precision metric classes $pm \in PM$. $unspecified \in PM$ denotes the precision metric class with the special meaning that no further specification is given for the interpretation of a precision metric. Each $pm \in PM$ denotes a logical expression composed of $p_{i.j} \in c_i$ or $pm \in PM$.*

Definition 3. *Function* pmetric*: $F \rightarrow PM$, with $pmetric(f) = pm$, returns the precision metric class $pm \in PM$ assigned to fact $f \in F$.*

No precision metric classes are specified at the very beginning, i.e., for all $f \in F$, $pmetric(f) = unspecified$. Precision metric classes will be dynamically derived from the combination of similarity predicates $p_{i.j} \in c_i$ created by users over time, by means of crowdsourcing (or over a training phase used to calibrate the system). If users state that values are consistent although they are not literally identically (i.e., not of predicate $p_{1.1}$), the appropriate precision metric class can be determined by means of "learning" from the user's specifications.

Expert users of the system can choose between using the system's suggested precision metric classes or defining and providing their own set of precision metric classes or even precision functions when elaborating information on the Web or making use of a common selection of *FactCheck* precision metric classes. Depending on the user's choice the output of the function *check* (see Definition 4 of two fact values can be defined as non-conflicting or conflicting.

Definition 4. *Function* check*: $V \times V \times PM \rightarrow \{non\text{-}conflicting, conflicting\}$, with*

$$fcheck(v_1, v_2, pm) = \begin{cases} non\text{-}conflicting, & \text{if the logical expression of } pm \\ & \text{evaluates to true for } v_1 \text{ and } v_2, \\ conflicting, & \text{otherwise.} \end{cases}$$

The definitions above are used to describe the formalized information model underlying *FactCheck* by use of set theoretic semantics. The formalization covers the central concepts of the *FactCheck* model. Formal definitions regarding the interaction of *FactCheck* with the user crowd are not needed for detailed understanding of the model and are not further addressed here. These semantics are later used to explain the architecture and the prototypical implementation of *FactCheck*.

3.2 System Architecture

FactCheck consists of five major components: (i) FactBase, which stores all facts and meta information of processed sites (like a data cash growing over time), (ii) FactServer, which actually does the comparison of facts, (iii) FeedbackBase, which is needed to save the user's markups for conflicting data, (iv) Precision Metric Manager, which is needed to incrementally build up and adapt the precision metric classes, and (v) Information Dashboard, which is needed for information, management, and notification of owners of data sources about conflicting data related to their content.

These architectural components are designed such that an implementation integrated with existing web-infrastructure components as existing web servers and browsers can be realized, envisioning the FactCheck functionality to become an integral part of a future Web-IT-infrastructure. Due to space limitations further details cannot be presented in this paper.

4 IdaFix - An Implementation of FactCheck

For the purpose of demonstration and evaluation we developed a prototypical *FactCheck* backend infrastructure consisting of a FactServer, FactBase, Precision Metric Manager and Information Dashboard. In order to allow users to interact with *FactCheck* we developed *IdaFix*[7] as the *FactCheck's* frontend. *IdaFix* is a Web application with the focus to allow a crowdsourcing reduction of conflicting data on the Web.

Currently *IdaFix* supports four use cases: (i) notify a user about potentially conflicting data on the Web page the user is visiting, (ii) as a next step, offer four lists of equal, conflicting, locally missing and remotely missing facts and provide a feedback interface, to let the user remark whether the proposed conflicts are assigned to the right list from his point of view, or not, (iii) trigger a notification to data source owners about conflicting or missing data on their sites.

All components of *FactCheck* are implemented in Python. To ease the process of entity reconciliation FactServer operates with Google's machine identifier by using the API of Google Knowledge Graph[8]. The cleaning and normalization module is well trained in the domain of movies, books, persons, product reviews, and product facts, but can also be used in other domains.

Figure 1 shows the small information box which appears in the browser when visiting a Web page using *IdaFix*. In Fig. 2 an example of the conflicting tab is presented with two conflicting facts.

[7] The name *IdaFix* is derived from the main purpose of *FactCheck* to **Id**entify **a**nd **fix** structured data on the Web.

[8] https://developers.google.com/knowledge-graph/.

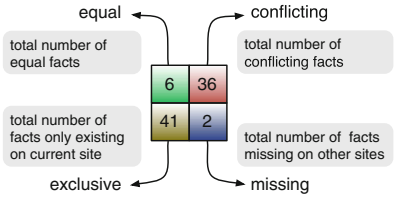

Fig. 1. *IdaFix's* summary of facts

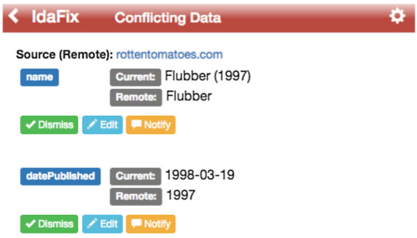

Fig. 2. Exemplary output of browser extension *IdaFix*

5 Discussion

To evaluate a part of the proposed framework, we surveyed 80 persons in 6 scenarios about the impact of the information indicators provided by *IdaFix*. We evaluated the assessment of positive and negative effects of the additional knowledge *IdaFix* users may consume. Due to space limitations further details on the promising evaluation results cannot be presented here. In addition we implemented a proof of concept implementation of the framework consisting of a FactServer, the IdaFix add-on, the Information Dashboard, the Precision Metric Manager and a FactBase with currently holding more than 10 million facts.

6 Conclusion

In this paper we describe a model for a Web infrastructure, which provides built-in measures

(i) to reason about the quality of structured data,
(ii) to detect conflicting pieces of information on the basis of analysis and comparison of structured data,
(iii) to allow for user- or crowd-based indication and annotation of potentially conflicting data,
(iv) to support the correction or resolution of conflicting data by offering automatically generated correction or resolution proposals to data owners and data sources, even to content management tools.

Additionally, we defined our concrete framework *FactCheck* based on the general idea and approach in the specific technical context of Microdata and JSON-LD data on the Web and allow users to identify and fix structured data encoded by these formats. Furthermore, it allows detailed analysis of inconsistencies on the Web that can be used for conclusions about websites. The impact of the information indicators on websites was evaluated and verified in a survey proving that *IdaFix* to a large extent influences the perception of websites and is having an impact to users on various different tested levels and usage scenarios.

In our future work, we aim to extend our model and the prototypical implementation of our model. Taking into account advanced methods of content analysis tools like NLP or image analysis, *FactCheck* does not need to be limited to structured data elements embedded in Web pages, but will also be applied to any type of content on the Web.

References

1. Adida, B., Birbeck, M.: RDFa primer 1.0: embedding RDF in XHTML. W3C Working Draft (2006)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Web Semant. **7**(3), 154–165 (2009)
3. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. **41**(1), 1–41 (2008)
4. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 601–610. ACM (2014)
5. Goel, K., Guha, R.V., Hansson, O.: Introducing rich snippets. Google Webmaster Central Blog (2009). https://goo.gl/e8P4fs
6. Guha, R.V., Brickley, D., Macbeth, S.: Schema.org: evolution of structured data on the web. Commun. ACM **59**(2), 44–51 (2016). https://doi.org/10.1145/2844544
7. Haas, K., Mika, P., Tarjan, P., Blanco, R.: Enhanced results for web search. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 725–734. ACM (2011)
8. Hassanzadeh, O., Pu, K.Q., Yeganeh, S.H., Miller, R.J., Popa, L., Hernández, M.A., Ho, H.: Discovering linkage points over web data. Proc. VLDB Endow. **6**(6), 445–456 (2013)
9. Hickson, I.: HTML Microdata. World Wide Web Consortium (2011). http://www.w3.org/TR/microdata
10. Isele, R., Bizer, C.: Learning expressive linkage rules using genetic programming. Proc. VLDB Endow. **5**(11), 1638–1649 (2012)
11. Khalili, A., Auer, S.: WYSIWYM authoring of structured content based on schema.org. In: Lin, X., Manolopoulos, Y., Srivastava, D., Huang, G. (eds.) WISE 2013. LNCS, vol. 8181, pp. 425–438. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41154-0_32
12. Meusel, R., Petrovski, P., Bizer, C.: The WebDataCommons Microdata, RDFa and Microformat dataset series. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 277–292. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_18
13. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2016)
14. Schultz, A., Matteini, A., Isele, R., Mendes, P., Bizer, C., Becker, C.: LDIF-a framework for large-scale linked data integration. In: 21st International World Wide Web Conference (WWW 2012) (2012)
15. Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., Lindström, N.: JSON-LD 1.0: a JSON-based serialization for linked data. W3C Recommendation (2014)
16. Tayi, G.K., Ballou, D.P.: Examining data quality. Commun. ACM **41**(2), 54–57 (1998)

17. Taylor, N.E., Ives, Z.G.: Reconciling while tolerating disagreement in collaborative data sharing. In: SIGMOD Conference, pp. 13–24. ACM (2006)
18. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
19. Wang, R.Y., Strong, D.M.: Beyond accuracy: what data quality means to data consumers. J. Manag. Inf. Syst. **12**(4), 5–33 (1996)