universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„FactChecking - Towards conflict analytics (dashboard) for content providers“

verfasst von / submitted by

Jakob Hirschl, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066935
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Medieninformatik
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Klas

Acknowledgments

First of all I want to thank my supervisor Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Klas for his excellent guidance. He was always available during the whole process to give valuable feedback, even extending meeting times to late hours if necessary.

I am no less thankful for the support of Dipl.-Ing. Dr. Peter Kalchgruber, especially for helping me understand the already existing components of the FactCheck++ framework.

I also want to thank all fellow students who participated in the FactCheck++ workshops which was helpful for understanding the bigger picture and get a fresh view on some aspects.

I would also like to thank my father Univ.-Prof. Dr. Alexander Hirschl for second reading the thesis and my wife Amanda de Paula Cristino Hirschl BSc for enduring lengthy discussions and helping me with her own expertise when I got stuck.

Last but not least I want to thank my whole family for their support and encouragement throughout the process of writing this thesis.

Abstract

Despite its widespread adoption, conflicts in structured data on the web can still occur and lead to the spread of misinformation. The rise of the Internet and the popularity of social media made it easy for misinformation to spread quickly and has shown to be a threat to the well-being of the society. While fact-checking organizations rise to combat this, the amount of misinformation is too large to handle, which led to the development of (semi-) automatic fact-checking methods. Data produced by this method usually require proper analysis and visualization of the results to make them understandable for end-users. While many such examples can be found for researchers and content consumers, content providers are rarely considered, although they have an essential role in fighting misinformation, being the ones who can prevent it before it can even spread. Providing them with metrics and visual tools to check their content could greatly benefit the combat against misinformation.

This thesis presents various metrics that can be useful to content providers to assess their content in addition to a prototype of a dashboard, the Analytics Dashboard, that aims to support content providers in keeping structured data free of misinformation. The dashboard will be part of the FactCheck++ framework, which can detect and resolve conflicts in structured data on the web. It will use the data provided by the framework to calculate various metrics and present them in a practicable way to content providers. Finally, it is shown how the prototype can be used to find and support fixing potential sources of misinformation. The benefit of this approach in the fight against misinformation will be discussed.

Kurzfassung

Trotz der weiten Verbreitung von strukturierten Daten im Web, treten zwischen diesen noch immer Konflikte auf, die in Folge zu Verbreitung von Missinformationen führen können. Der Aufstieg des Internets und die Popularität der sozialen Medien haben die schnelle Verbreitung von Missinformationen erleichtert, was sich als Bedrohung für das Wohl der Gesellschaft erwiesen hat. Faktencheck-Organisationen versuchen dem entgegenzuwirken, jedoch ist die Menge an Missinformation zu groß um damit umzugehen. Dies war der Anlass zur Entwicklung (semi-) automatischer Faktenchecking-Methoden. Mit diesen Methoden erzeugte Daten erfordern jedoch eine Analyse und Visualisierung der Ergebnisse, um diese Daten für Endbenutzer verständlich zu machen. Während für Forscher und Inhaltskonsumenten viele solcher Beispiele zu finden sind, werden Inhaltsanbieter selten in Betracht gezogen, obwohl sie eine wesentliche Rolle bei der Bekämpfung von Missinformation haben. Sie können nämlich Missinformation verhindern, bevor diese sich überhaupt verbreiten kann. Die Bereitstellung von Metriken und visuellen Tools zur Überprüfung ihrer Inhalte könnte bei der Bekämpfung von Missinformation von großem Nutzen sein.

Die vorliegende Arbeit stellt verschiedene Metriken vor, die für Inhaltsanbieter zur Bewertung von Inhalten nützlich sein könnten. Außerdem wird der Prototyp eines Dashboards präsentiert, das Analytics Dashboard, welches Inhaltsanbieter dabei unterstützen soll, strukturierte Daten frei von Missinformation zu halten. Das Dashboard wird Teil des FactCheck++-Frameworks sein, das Konflikte in strukturierten Daten im Web erkennen und lösen kann. Vom Framework bereitgestellten Daten werden dazu verwendet, um verschiedene Metriken zu berechnen und diese für Inhaltsanbieter geeignet darzustellen. Letztendlich wird gezeigt, wie der Prototyp verwendet werden kann, um potenzielle Missinformationsquellen zu finden und zu beheben. Der Nutzen dieses Ansatzes bei der Bekämpfung von Missinformation wird diskutiert.

Contents

List of Tables

List of Figures

1. Introduction

1.1. Motivation

Structured data, used by search engines to provide improved search results, can be found on various websites, and it is expected that its amount will further increase [34]. In the course of their work, Kalchgruber et al. [34] found that, despite its widespread adoption, information is not always consistent or valid, leading consumers to draw false conclusions. The resulting misinformation can spread and affect many domains, including economy, health, and politics [19].

During the recent corona crisis, a joint statement by world organizations emphasizes how misinformation can cost lives, polarize the public debate, and not only threaten the combat against the virus but also heighten the risk of conflict and violence [33]. Roozenbeek et al. [55] found that a substantial proportion of their sample across different countries views misinformation about the virus as highly reliable. Furthermore, a small group regards factual information as unreliable, negatively affecting self-reported compliance to public health guidance. In a recent survey, Zhou and Zafarani [78] even stated that fake news is *"one of the greatest threats to democracy, journalism, and freedom of expression"*.

Although misinformation is not a new phenomenon of the internet [75], its increased use as an information source [57] and the popularity of social media, with its lack of oversight [58], has proven to be a fertile breeding ground for misinformation [74, 19]. Even though studies show that false rumors tend to correct themselves [12, 50], it can take time until this happens [79] since fact-checking content spreads at a slower rate than misinformation and therefore tends to lag behind [19]. Political and economic benefits of spreading disinformation also motivate the use of social bots, which are computer algorithms that create content and interact with social media users, to speed up the spread of misinformation [20].

The spread of misinformation led to an increase in fact-checking [38], but the manual process is time-consuming, expensive [44, 19], and cannot keep up with the amount of created misinformation [78]. Automatic and semi-automatic fact-checking methods try to overcome these issues to support fighting misinformation (e.g. [11, 31, 73]). They rely on different techniques like natural language processing [44, 13], information retrieval [13], data knowledge management [13, 76], machine learning, network/graph theory [78], or crowd signals [69] to find misinformation. While being more efficient, they are less precise than manual approaches, hence proper analysis and visualization of the results and an explanation why the algorithm made certain decisions are required to help users make a profound decision about the content.

Not only researchers and content consumers but also content providers and social media platforms are interested in keeping content free of misinformation [42, 17], as can be seen in the effort of Twitter and Facebook during the 2020 US Election to delete or mark misleading statements on their platforms. Twitter also recently started the test phase for a crowd-based fact-checking project, where a community of volunteers can identify and add context to misleading content [15]. While a study by Allcott et al. [2] and statistics by Resnick et al. [53] show that attempts by content providers to reduce misinformation might be effective, the 2020 US presidential election again brought various examples of misinformation spreading via the social media platform and criticism that the measures taken did not go far enough [37, 54, 39].

Once lies are spread, they often get more attention than fact-checking organizations [38] or major news outlets. This could be observed in the upcoming months of the 2016 US presidential election. The top 20 performing false elections stories generated more engagement than the top 20 best performing election stories from major news websites [62]. The most efficient way to avoid the spread of misinformation is to detect and correct it as soon as possible [74, 19].

In that respect, content providers play a crucial role. By keeping their content free of misinformation, they can stop it before it has the chance to spread. This motivates the development of a dashboard that analyzes and visualizes data about misinformation, like conflicts in structured data, directly targeted at content providers to support them in keeping content consistent and offering them metrics to evaluate their progress. Many examples of how this data can be analyzed and visualized exist (e.g. [51, 52, 45, 31, 73, 6]), but they usually target content consumers and researchers rather than content providers. While some interests of those groups overlap, content providers require more detailed information about their content. The main interest of content consumers might be to see if the content (or a content provider) is reliable. At the same time, researchers might want to get an overview of a specific topic rather than all content of a single provider. Content providers, on the other hand, need to know what exactly is wrong with their content so that they can fix it.

1.2. Research Question

The question of this thesis is how data about misinformation can be analyzed and visualized to support content providers in avoiding misinformation on their platforms? It can be subdivided into three research questions:

1. What metrics are useful for content providers, and how can they be gained from data about misinformation? Raw data alone might not be sufficient in many cases. The goal is to find and create various metrics calculated from data often available regarding misinformation.

2. How can those metrics be presented in an easily understandable way for content providers? Simply showing several graphs with all the different calculated values would not be sufficient. Both an overview of all content and detailed information concerning single content entities (e.g., an article or structured data about a movie) and a solution to navigate between them need to be designed.

3. How can the answers to the previous research questions be used to create a prototype based on the FactCheck++ framework? While the developed core concepts should apply to different data sources, this point aims to show their use in a concrete example.

Calculation or collection of misinformation data from content (articles, structured data, etc.) is not part of this thesis. All concepts presented assume that the data is already available and can be used to derive the values required for the metrics. An analysis of the validity of the algorithms creating the data is also not the subject of this thesis.

1.3. Outline

Chapter 2 will start with a short introduction to the topic of misinformation and fact-checking, followed by a description of the concept for the FactCheck++ framework, including essential terms in the project, and an overview of related examples analyzing and representing misinformation. Chapter 3 then presents the requirements derived from the research questions and a concept for metrics and the dashboard resulting from them. The implementation of this concept is shown in chapter 4. In chapter 5, the prototype is evaluated, and the limitations of the prototype and the FactCheck++ framework are discussed. Finally, chapter 6 will conclude the thesis and give an outlook on possible future work.

2. Related Work

This chapter will start with a short definition of terms for false information used in this thesis in section 2.1, followed by a brief introduction to the topic of fact-checking in section 2.2. Section 2.3 then presents the concept behind the FactCheck++ framework and lists several essential terms. Finally, section 2.4 will then present related applications for analyzing and representing misinformation.

2.1. Definition of Terms for False Information

False information can be separated into *misinformation* and *disinformation,* but determining what belongs in which category can be difficult. There are many interpretations of misinformation, but they are not always accurate [76], and what is considered misinformation can be politically disputed [53]. Wardle and Derakhshan [71] and Zubiaga et al. [79] refer to misinformation as false information that is shared unintentionally/without malicious intention, while disinformation intends to cause harm. Zhou and Zhang [76] also include *"information that does not reflect the true state of mind"* of an author in the definition of misinformation. Resnick et al. [53] take an aggregated recipient-centric approach to defining misinformation as *"information that most recipients would judge to be false or misleading after taking the time to carefully consider all of the evidence about it"*. Other sources refer to false information as misinformation in general or use both terms interchangeably (e.g. [58, 51, 31, 60]).
In this thesis, the term misinformation will refer to incorrect information regardless of the intent, while disinformation is used for incorrect information where there is a clear malicious intention since this is what most sources seem to agree on.

There are many ways to subdivide disinformation that specify their goals and means of propagation. *Fake news*, for example, describes a false news story published via a news outlet to reach a particular malicious goal [78, 32, 63]. The term fake news is not only widely used in research (e.g. [78, 73, 10]) but also popular as a synonym for (not always) false information to the point that some argue it is overused and alternative terms should be used [70, 75]. Another category worth mentioning is *satire*, which describes false information, not with the intent of misleading but to entertain the reader [5, 63]. Because of its potential to fool [70], it is by some (arguably incorrectly) classified as fake news and the difficulty to recognize it even by human readers [5] makes it a challenge for fact-checking.
Staff [63] and Wardle [70] present various further categories that would go beyond the scope of this thesis.

Lastly, *rumors* should be mentioned, which are not necessarily false information but only information that cannot be verified at the time of its creation [79]. Still, there are many similarities between rumors and false information [79], so research in this area is also relevant in this thesis.

2.2. Fact-Checking

2.2.1. Types of Fact-Checking Techniques

Expert-based manual fact-checking is done by news organizations and various specialized fact-checking organizations like Snopes [28], PolitiFact [49], FactCheck [18], or NewsGuard [66]. It is a time-consuming and expensive process [44, 19] that cannot keep up with the massive volume of misinformation created [78]. It often includes small groups of experts checking news articles, complying with a code of principles (e.g. [48]). The results produced by those organizations are highly accurate and can potentially be used as ground truth for comparative studies [78].
Another manual approach other than this expert-based approach is the *crowd-sourced manual fact-checking*, which relies on a large population of regular individuals gathered on crowd-sourcing marketplaces like Amazon Mechanical Turk [78]. As Pennycook and Rand [46] argue, those approaches may have an advantage since the crowd can find not only factually wrong but also biased content. Compared to the expert-based approach, crowd-based approaches are less credible, less accurate, and still not sufficiently scaling to the volume of misinformation [78].

Automatic fact-checking addresses the issue of scalability from manual fact-checking by relying on techniques like natural language processing [44, 13], information retrieval [13], data and knowledge management [13, 76], machine learning, or network/graph theory [78]. Castillo et al. [11] analyzes the credibility of news on Twitter using various machine learning techniques. While they are not directly handling content's veracity, such work is often considered a reference [79]. Gerber et al. [23] developed *DeFacto*, an algorithm for validating RDF triplets by finding confirming evidence collected from web pages written in several languages. It also supports facts with a temporal scope, i.e., facts that are only valid for a specific time frame. Yu et al. [74] propose a Convolutional Approach for Misinformation Identification based on Convolutional Neural Network and show promising results for misinformation identification and early detection. Aker et al. [1] and Kochkina et al. [36] utilize stance classification, determining the attitude of the author, which they consider to be an important step towards rumor verification. Often automatic methods still rely on expert-based manual fact-checking for their findings (e.g. [69], [31]), but as Tschiatschek et al. [69] already noted, computational techniques should in general complement, but not replace expert validation.

Similar to Fernandez and Alani [19], this thesis considers *semi-automatic fact-checking* besides automatic fact-checking. They support fact-checking but rely on manual labor for more than, e.g., just a final fact check by experts. Semi-automatic fact-checking includes tools supporting human fact-checkers in finding content that might be worth checking or ease their task of researching claims. A tool presented by Resnick et al. [52] helps journalists to identify rumors on Twitter and supports them in investigating their truthfulness and diffusion. The approach by Metaxas et al. [40] has a similar goal to help journalists to find the origin and spread of tweets. Another example is DETECTIVE [69], which utilizes crowd signals to select a small subset of news that experts can then fact-check. The unreliability of the crowd is handled by implementing a Bayesian approach to learn about users' accuracies over time. Facebook is using a similar approach allowing its users to flag fake content [42].

For this thesis, automatic and semi-automatic approaches are especially interesting since they often analyze and present their results for different target groups (e.g. [31, 52, 73]).

For more examples of the current state-of-the-art in automatic (and semi-automatic) fact-checking, the survey paper of Zubiaga et al. [79] gives an overview with a focus on identifying rumors on social media. Oshikawa et al. [44] and Guo et al. [30] focus on methods using natural language processing. Finally, Zhou and Zafarani [78] give an overview on the topic and methods to detect *fake news*.

2.2.2. Challenges of Fact-Checking

Various challenges are still open in the field of fact-checking [78, 44, 41, 13, 19, 30]. The recent survey of Zhou and Zafarani [78] suggests that there are open challenges in detecting non-traditional fake news, like outdated facts or partially false content, and improving early detection and intervention. They also suggest that the focus should shift from single social networks or languages to cross-domain (-topic, -website, -language) analysis. Both Zhou and Zafarani [78] and Mohseni et al. [41] see the explainability of algorithms used for detection as a future challenge. Mohseni et al. [41] additionally mention improving human interpretability as a benefit which *"provides decision-making transparency to end-users with understandable explanations of "how the system works" and "why this decision is made""*. This challenge is especially relevant when presenting data for content providers, who are not necessarily fact-checking experts. Oshikawa et al. [44] consider the categorization in "true" and "false" as unpractical and suggest working on approaches that can categorize more fine-grained. Another difficulty described by Cazalens et al. [13] is that claims need to be put into perspective. A factually accurate claim can still be used in a misleading context, and identical facts have to be interpreted differently depending on the surrounding context. They also see the need for more standardization, explainability, and transparency of automated fact-checking tools [13].

2.3. FactCheck++

Today's Web pages contain structured information through semantically-rich embedded data more frequently. Search engines use this structured data to provide an improved semantic, structured search result. The publishing of semantically-rich embedded data was simplified by the introduction of various semantic markup capabilities. Widely adopted formats are Microdata, RDFa, Microformats, and JSON-LD, an extension to JSON. In addition, the introduction of schema.org further advanced its development. Bing, Google, and Yahoo created a standard set of schemas for various topics to allow user-friendly handling of structured data markup for content providers. However, despite its widespread adoption following drastic growth rates, structured data is often conflicting on different web pages. Those conflicts also affect applications and services that are built upon this data. An example of this is when Google Search falsely identified the loser of the second ballot as the new Austrian president [56]. While data fusion addresses this problem by cleaning, aggregating, and integrating data of an entity available from various data providers into a single clean, consistent representation, these algorithms can fail. They also do not address the problem of fixing the conflicting data at the source.

As a potential solution, the FactCheck++ framework [35] was proposed to detect and resolve conflicts in structured data on the web and confirm correctness or provide accurate data to rectify defective structured data. The framework offers three benefits: (i) it initiates and assists to solve conflicts directly at the data source, (ii) does not require complex data fusion algorithms, and (iii) helps improve the consistency of the web, implicitly allowing applications to be built upon more accurate data. While initially only applied to the domain of textual data, it is believed that the model can be extended to other domains like images.

2.3.1. Problem Formalization

FactCheck++ detects conflicts in structured data by comparing all representations (called instances) from different data sources (e.g., IMDb and RottenTomatoes) of a real-world named entity (e.g., the movie RoboCop). Each of those instances is composed of facts that carry a value from a set of valid values for the given fact (e.g., "DateTime" for "datePublished"). The values of the same fact of all instances can then be compared. Each comparison results in a confidence probability that expresses the likelihood of the two fact values being the same. The realization of this comparison can be manifold. It can reach from only using the value-pairs to a comparison of richer contextual information, similar to current cognitive computing services (e.g., those used by the IBM Watson services, Microsoft Cognitive Computing Services, Google Services, etc.). The proper notion of similarity in structured data is crucial since various reasons for different fact values (e.g., technical restrictions, different times of measurement, etc.) exist. However, those differences would not always be considered a conflict by an observing user. For example, considering stock prices, a slight deviation might be regarded as a conflict, while minor deviations in a city's population might be regarded as equal. The similarity between two fact values always depends on the observer or use case of the data. Therefore

facts are classified using different similarity predicates, determining if two given values are conflicting or matching (equal/non-conflicting). To handle the potentially large variety of predicates, the concept of precision metrics was introduced. They allow the smart combination of similarity predicates by logical operators. For example, a fact (e.g., "datePublished") can be assigned to a precision metric class, consisting of one or more similarity predicates, dynamically over time utilizing user feedback.

2.3.2. FactCheck++ Architecture

Figure 2.1 shows the current architecture of the framework.

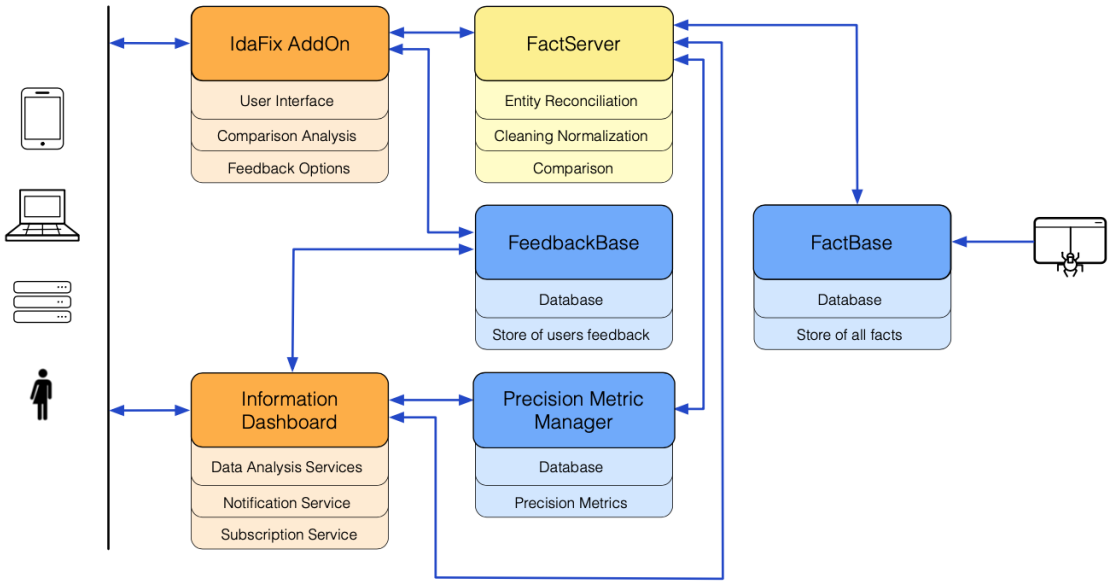

Figure 2.1.: The system architecture of FactCheck++ (components and information flow) [35].

- *FactBase:* This is the central storage for instances that were gathered. It can be populated and maintained by three different methods: (i) by adding already existing collections, (ii) by web crawlers gathering structured data, and (iii) by the IdaFix plugin used by clients. The FactBase also saves all statistical information about the instances. It is primarily used by the FactServer to support the reconciliation and comparison of instances.

- *FactServer:* This is the central component of the framework. It (i) offers an API to retrieve data about instances and (ii) offers the possibility to compare instances with the existing FactBase. If the compared instance is not in the FactBase or has changed, its values and statistics will be updated. In addition, all statistics of related instances will be updated, therefore populating and maintaining the FactBase. The most important sub-components of the FactServer are the *Entity Reconciliation* component, which is responsible to decides which named entity an instance belongs to, and the *Comparison* component, which is responsible for the actual comparison of facts.

- *FeedbackBase:* This component manages user feedback from all different front-end components of the framework. It can include both feedback of users assessing potentially conflicting facts and feedback about the components themselves (e.g., bugs or other issues with the UI). The information can then be used to improve the precision metrics used and further improve users' experience with FactCheck++.

- *Precision Metric Manager:* This component manages all the data and services related to precision metrics. It internally deals with the unsupervised learning to create new precision metrics or the adoption of existing precision metrics to improve the system behavior using the information provided by the FeedbackBase. It also allows to request information about precision metrics and manage precision metrics manually.

- *Information Dashboard:* This component is the interface for expert users, like content providers, researchers, or even applications, to the FactCheck++ framework. It utilizes the information provided by the FactServer and offers multiple services to consume them. The *Data Analysis Services* consist of an API to retrieve everything from individual conflicts to aggregated information on domains or instances. The service also offers user interfaces to explore this information.

- *IdaFix plugin:* IdaFix (derived from "identify and fix structured data on the web") is the equivalent of the Information Dashboard for content consumers. The browser plugin extracts structured data from any website, sends it to the FactServer for evaluation, and displays the comparison results in a way easy to understand for content consumers. Additionally, it populates and maintains the FactBase by requesting comparisons for new or updated instances. It will also be possible to give user feedback via IdaFix.

This thesis is primarily placed in the Data Analysis Service of the Information Dashboard component. It will add a new service allowing content providers to explore the information provided by the FactCheck++ framework. Since the metrics calculation will require data only available during the comparison of facts, it will also be necessary to change the Comparison component of the FactServer. Finally, the newly calculated values need to be saved, which will require changes in the FactBase, which is responsible for the persistent storage of the statistics.

2.3.3. Information Dashboard UI

The FactCheck++ framework already has a UI for the Data Analysis Services, but it does not fulfill the goal of this thesis to support content providers in keeping their content consistent. It is possible to see information about the state of the domain (and also to compare multiple domains). However, there is no possibility to infer the instances responsible from the aggregated statistics (cf. figure 2.2).

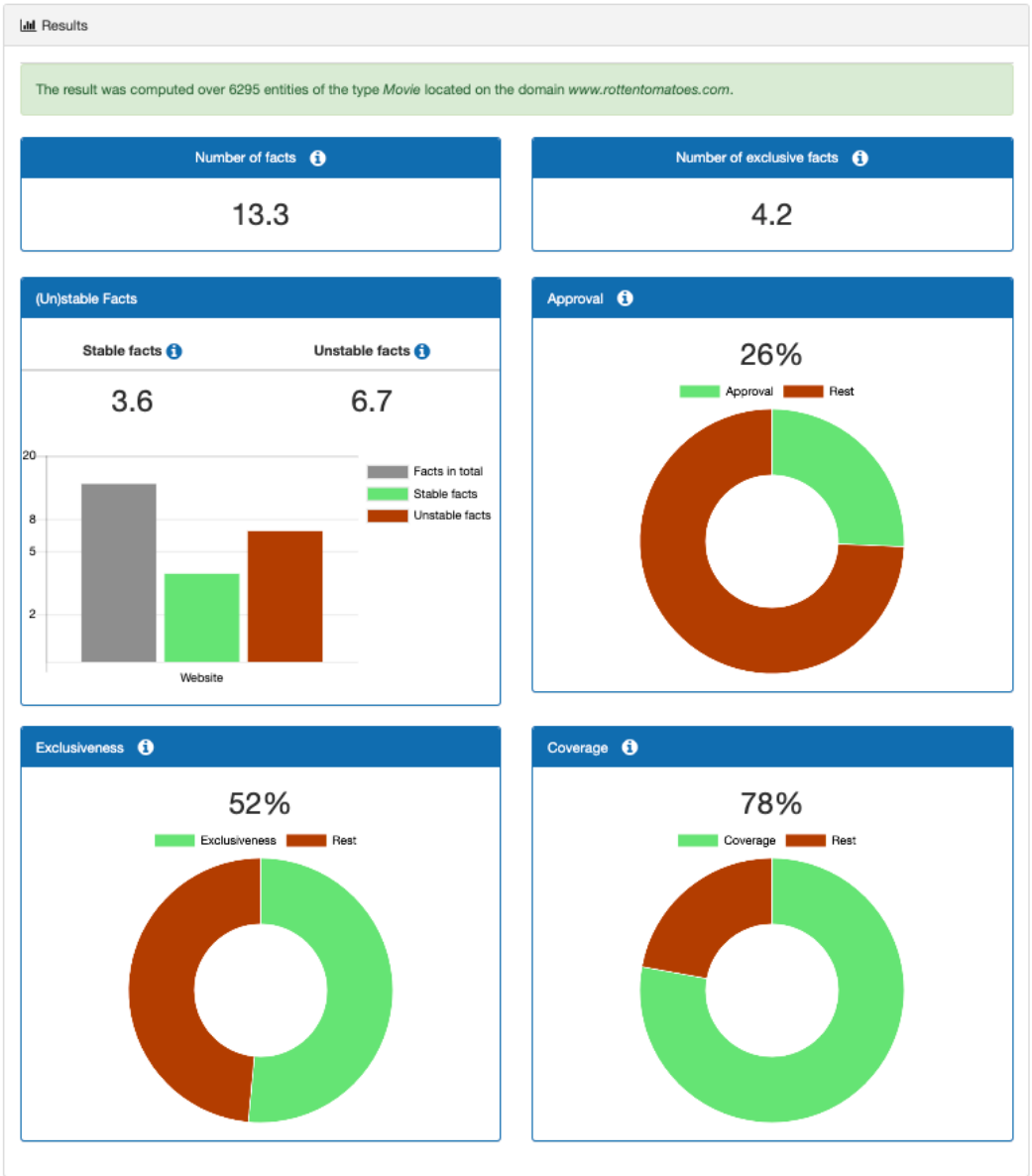

Figure 2.2.: The Information Dashboard UI showing the aggregated data for rottentomatoes.com.

Looking at the approval value of 26% in figure 2.2 it is impossible to know if this is caused by a small number of instances with low approval or if all instances on the domain have a low approval. The dashboard gives content providers an idea about the state of their domain but does not support them in fixing any potential issues. This would require a connection to single instances.

Assuming that they still manage to find a specific instance that, for example, has a low approval, they will be shown an almost identical UI. It can help assess whether an instance should be checked or not, but they again do not assist in finding the cause for any results. The information about the instance's multiple values is aggregated, thus it is impossible to say which is matching or conflicting.

In conclusion, the existing UI can be useful for researchers who want to get an overview of the data collected by the FactCheck++ framework. For content providers, on the other hand, it is not detailed enough to enable them to increase the consistency of structured data on their domain.

2.3.4. Important Terms

This section lists several terms used in the FactCheck++ framework that are important to understand this thesis.

- *Entity:* An entity is a self-contained thing that can be described with facts. They are concepts or phenomena and may be fictional or non-fictional. *Named entities* can be represented with a name (i.e., an URI) and have a physical or abstract existence. Examples can be the movie *RoboCop* or the person *JRR Tolkien*.

- *Fact set:* The fact set is a collection of unique facts about an entity, i.e., it is the union of all facts that instances about this entity contain.

- *Instance:* An instance is the representation of an entity on a specific domain. It consists of several fact values. It does not necessarily have a value for every fact of the entity's fact set. An example for an instance is the movie *RoboCop* on *rottentomatoes.com*.

- *Fact:* A fact is a piece of information about an entity. It can be thought of as a key-value pair with the key describing the type of information the value in an instance has. An example of a fact is "datePublished" with the value "1988-01-07". A fact of the observed instance is called a *local fact*, while a fact of any other instance is referred to as a *remote fact*.

- *Comparison results:* For every comparison of a fact from an instance (called the local instance) to another instance (called the remote instance), there are four possible results:
  - *Equal:* the facts are considered equal under the premise of a given precision metric (also called match or non-conflicting).
  - *Conflicting:* the facts are considered conflicting under the premise of a given precision metric.
  - *Locally missing:* A fact that exists in the remote instance does not exist in the local instance.
  - *Remote missing:* A fact that exists in the local instance does not exist in the remote instance.

  These results are the basis for all other statistics about instances.

- *Approval:* The fact of an instance can match with the values of other instances and conflict with others. The approval is the percentage of other facts that the local fact agrees with, i.e., the ratio between equal comparison results to the total number of comparisons. If, for example, three other pages have the equal value "1988-01-07" for the fact "datePublished" and one additional page has the conflicting value "1988-07-01", the approval is 0.75 or 75%.

- *Unique facts:* A unique fact is a particular case of remote missing where the fact only exists locally, i.e., it is missing in all other instances of the entity. It is currently not possible to know if a unique fact can be considered to be positive. On the one hand, it can contain rare and valuable information (e.g., new research results), but on the other hand, it can also contain false information or be assigned to the wrong fact (e.g., "publishedDate" instead of "datePublished"). They can be considered similar to a rumor in the sense that they are not necessarily false information but only information that cannot be verified at the time of its creation.

- *Freshness:* The freshness is a soft measure of how unique an instance of an entity is. It is calculated as the average ratio of remote missing facts to the number of local facts.

2.4. Analyzing and Representing Misinformation

Automatic (and semi-automatic) fact-checking helps keep up with the masses of misinformation on the internet, but the results produced by them alone are not always understandable for end-users. Pressured by heavy criticism, Facebook and Google started fighting the spread of misinformation on their platforms [19], making it essential for content providers to be more aware of this issue and rely on fact-checking themselves. Externally maintained metrics have the additional advantage of drawing attention to issues that are not prioritized and creating legitimacy for claims made about a platform [53]. Content providers can benefit from this. This chapter presents various approaches

that analyze and present data to fight misinformation and reviews if they, or parts of the approach, can be used by content providers to solve conflicting data.

2.4.1. Truthy

Truthy [51] was developed to track political memes on Twitter and help to detect misinformation in the context of US presidential elections. They collect data via the Twitter' gardenhose' API and scan them in real-time to identify tweets that have content related to the political elections and are of general interest. For this purpose, the scan uses a hand-curated set of keywords. A second filter then identifies memes that are considered to be of general interest. Their web interface allows users to explore the database and inspect individual memes. The visualization shows data like the number of tweets per user, the number of retweets, or the number of meme injection points, as well as temporal data, the diffusion network, and geo-locations of tweets (cf. figure 2.3).

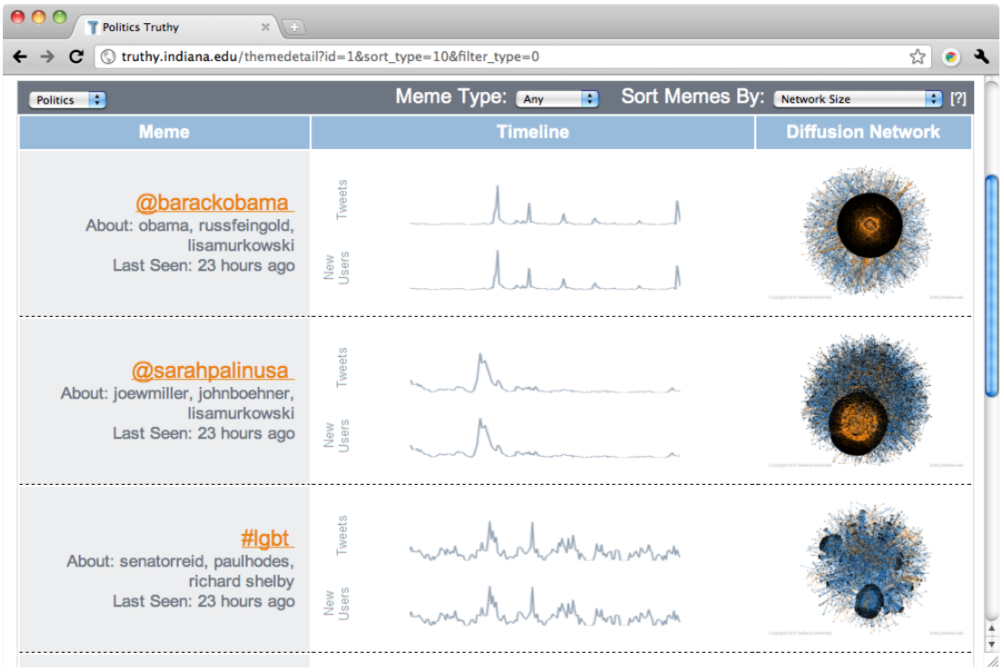

Figure 2.3.: Screenshots of the Truthy meme overview page [51].

The analysis and visualizations are very focused on the social media platform Twitter. It is unlikely that Truthy can help resolve conflicting data, but it shows how valuable temporal data can be to analyze the effect of content.

2.4.2. RumorLens

RumorLens [52] is a suite of interactive tools to support journalists in identifying new rumors and assess whether a rumorous content is interesting enough to justify further investigation. Tweets belonging to a rumor are collected and labeled as spreading or correcting. For the visualization, they treat user states, like the exposure to a rumor or active involvement, as nodes and flows of people between those states as links where the width of the link represents the number of people. The data is then visualized using a Sankey state transition diagram to show the diffusion of the rumor and a network diagram that can be used to find influential spreading or correcting tweets (cf. figure 2.4).

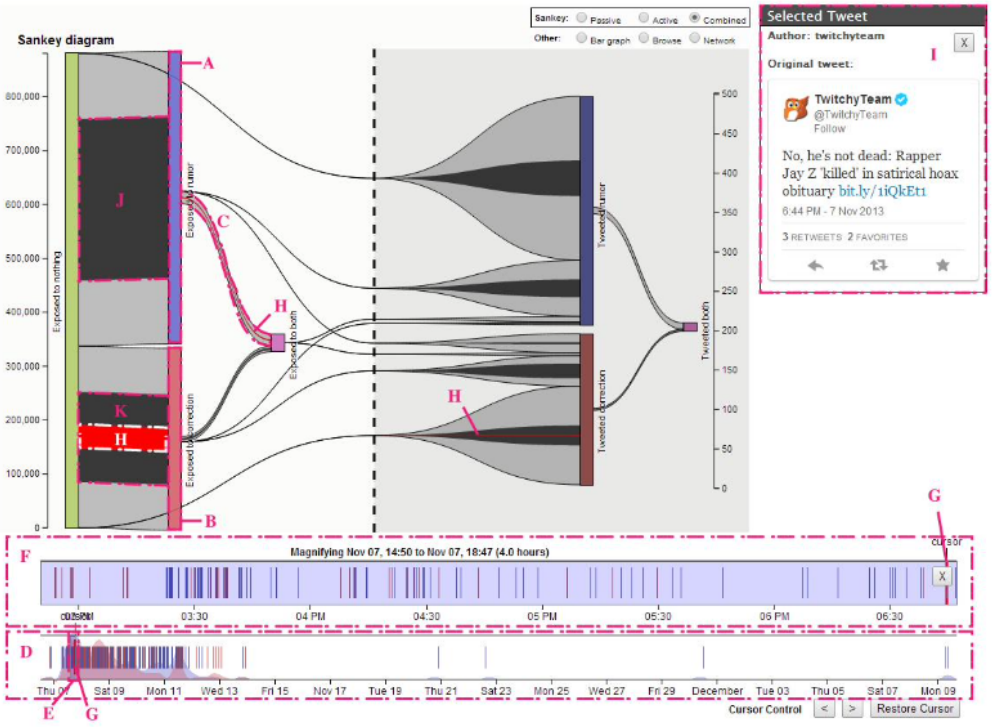

Figure 2.4.: Sankey state transition diagram of the rumor's diffusion [52]. A detailed description of the diagram can be found in [52].

While helpful to create new interesting content, the analysis and visualization are not useful to support content providers in fact-checking their own content.

2.4.3. Rumor Analysis & Visualization System

Pal et al. [45] propose a very similar approach to RumorLens, but they added the type of uncertainty-expressing messages to the already known types of rumor and counter-rumor. They used different visualizations to show the types, details, and trends of messages. Word clouds were used to show the most common words appearing in the different message types. A play track was embedded in a graph to show message trends. It visualizes the three types of messages over time with an additional panel below showing each tweet as a bubble with the size being determined by the author's number of followers (cf. figure 2.5). In addition, the different messages were color-coded to differentiate them. Also, a message dashboard was shown to see detailed information about single tweets.

Figure 2.5.: Visualizing Message Trends in [45].

Compared to RumorLens, the focus seems to lie more on researching rumor flow than using it to create content. While the analysis is also not helpful in analyzing conflicting data, the visualization of message trends could be an interesting approach to give an easy and visually appealing overview of the conflicting data in all created content.

2.4.4. ClaimBuster

ClaimBuster [31] is planned to be an end-to-end system using machine learning techniques, natural language processing, and database query techniques for computer-assisted fact-checking. It monitors different sources like websites, social, and broadcast media and gives each extracted sentence a score that describes the likeliness of being a claim. It then matches the claim against a repository or uses algorithmic fact-checking if no matching facts could be found. Results are tweeted via various channels.

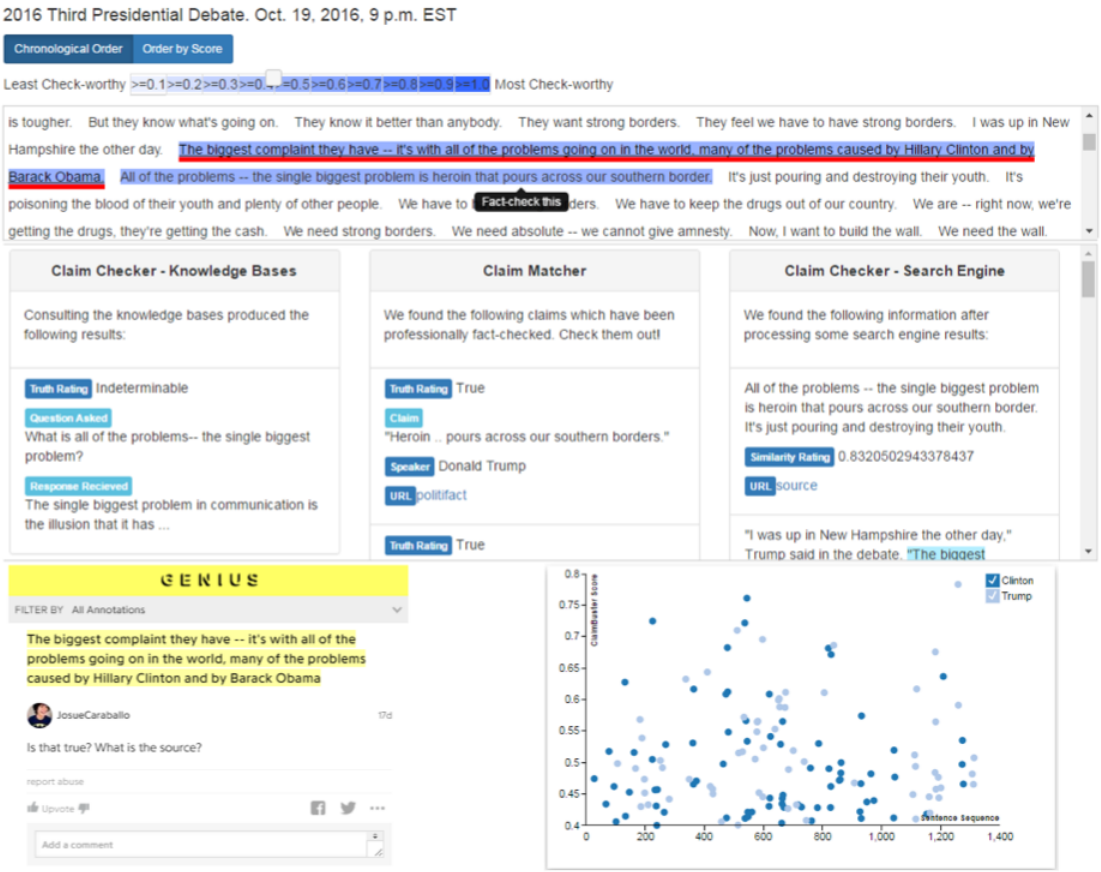

Figure 2.6.: The user interface of ClaimBuster when it is applied to a debate [31].

Most interesting for this thesis is the way they present their results by highlighting check-worthy claims with light or dark colors depending on the likeliness that it is a claim (cf. figure 2.6). Something similar could be helpful for content providers to easily see potential issues in the data that might be worth double-checking.

2.4.5. XFake

XFake [73] is an explainable fake news detector that assists end-users in identifying news credibility. For the effective detection of fake news items, they consider both news attributes, like the speaker or context, and statements in the news. Three different frameworks were designed for the analysis, each creating a prediction and explanation (cf. figure 2.7). The prediction of the system is the unified result from the framework, indicating the probability of the analyzed news being fake.

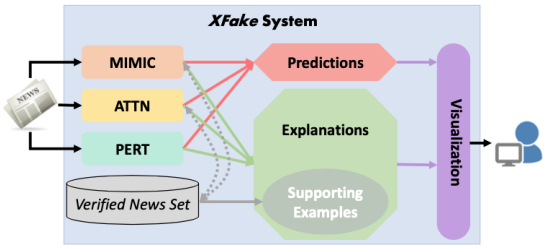

Figure 2.7.: The architecture of the XFake system [73].

The explanation differs between the frameworks and can be provided by key news components, word/phrase attribution, or linguistic features. Additionally, predictions are explained with generated supporting examples and further facilitated with a visualization of the prediction and the output. The visualization includes histograms for numeric values, highlighting important words with the darkness corresponding to the importance (cf. figure 2.8), and trees capable of showing the structure and activated paths of the model (cf. figure 2.9).

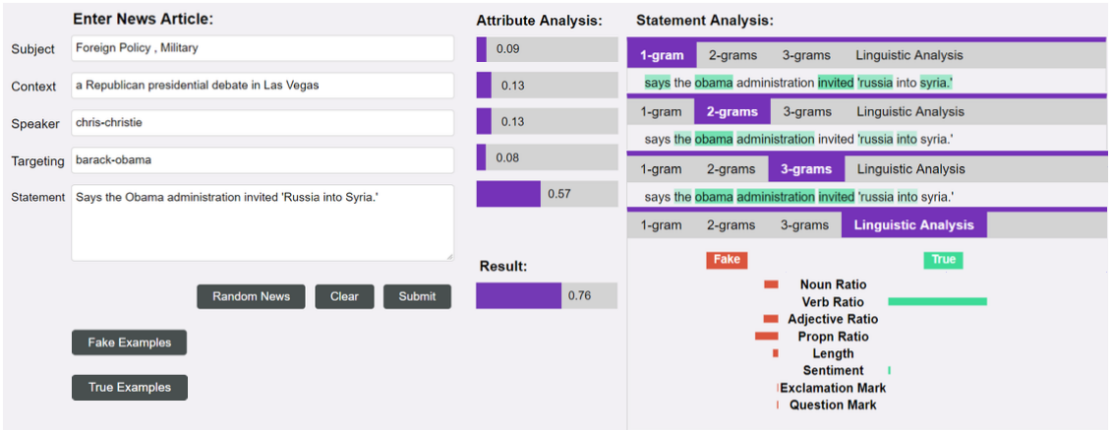

Figure 2.8.: Prediction and explanation from XFake [73].

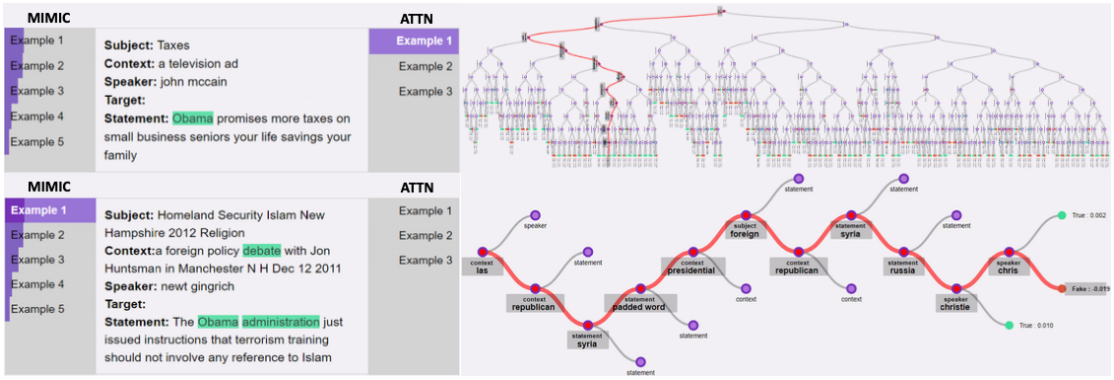

Figure 2.9.: Supporting examples and ensemble trees from XFake [73].

In addition to highlighting phrases already used similarly by ClaimBuster, the tree visualization providing better explainability of the model is especially noteworthy. While the visualization depends on the methods used for conflict detection and might differ from what is presented by XFake, this explainability would be helpful for content providers to understand results better.

2.4.6. Tweet Verification Assistant

Boididou et al. [6] present a system that can automatically classify multimedia Twitter posts as credible or misleading (content that does not faithfully represent the events it refers to). They extract credibility-oriented features from both the tweet and the author to train a two-step classification model to verify individual social media posts. The visualization shows a colored frame around the checked tweet and a bar below where green signals real and red fake content. It further allows inspecting the different feature values to get insight into the decision. When inspecting a value, a diagram for its distribution appears as well as a textual description telling the user the percentage of other tweets in this class (fake/real) that have the same value (cf. figure 2.10).

Similar to XFake, this is an excellent example of how the explainability of the decision can be presented. It also shows very well how detailed information about a single item can be presented in an easy to understand and intuitive way with a clear and straightforward rating for users just wanting an estimation of how reliable the content is, and detailed, less prominently placed information for those who want to dive deeper into the reason for the rating.

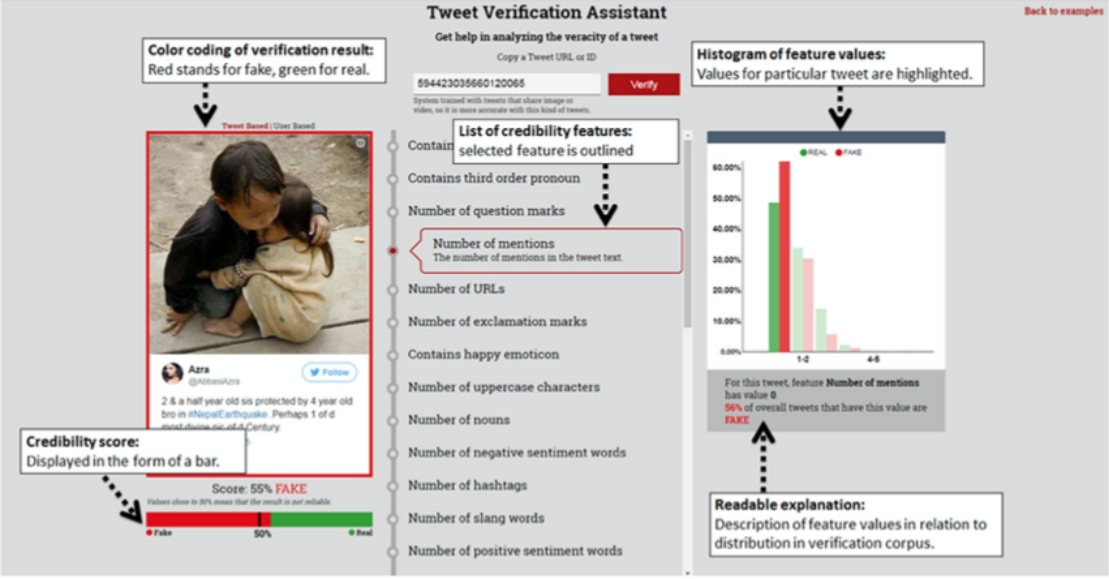

Figure 2.10.: Snapshot of the Tweet Verification Assistant interface [6].

2.4.7. Hoaxy

Hoaxy [58, 59] was first introduced in 2016 as a platform to collect, detect and analyze misinformation and its related fact-checking efforts [58]. Data was collected since then, and the results were presented in [59]. Other than previous approaches, the system monitors the social media stream automatically using keywords to filter it. Tweets that pass the filter are then checked for the occurrence of a URL to one of the news source sites that they observe (using RSS and web-crawling). Finally, all collected data is saved in a database. The provided web interface allows users to search and visualize the spread of claims and their competition with fact-checking verifications. Two types of visualizations are provided. A time plot displays the number of tweets for claims and fact-checking, and an interactive visualization shows the diffusion network.

As they state themselves, Hoaxy is similar to the previously presented RumorLens, and while there are significant differences, they are not relevant for this thesis. However, it is noteworthy that their system architecture is similar to the planned architecture of the FactCheck++ framework. Because of this similarity, the challenges they faced, like web content duplication, filtering relevant information from the site, or handling web crawling, potentially relevant for future work (cf. figure 2.11).

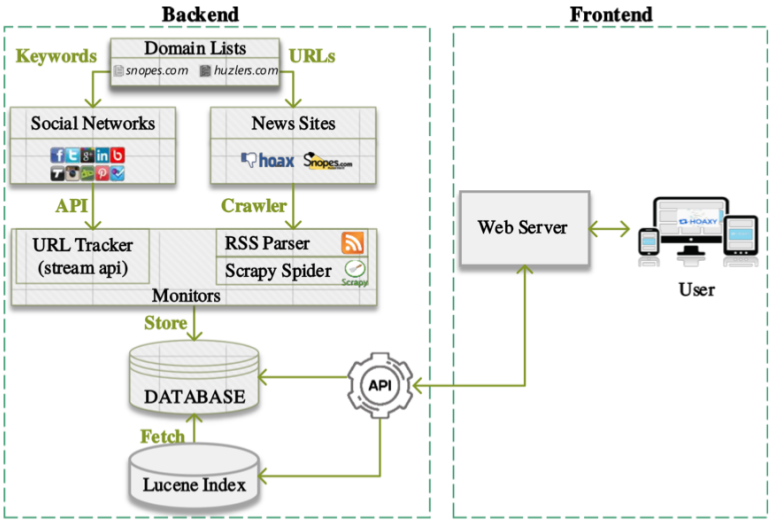

Figure 2.11.: Hoaxy system architecture [59].

2.4.8. "Iffy" Quotient

The "Iffy" Quotient [53] is a metric that describes the amount of "Iffy" content that is amplified on Facebook and Twitter. As "Iffy", they describe sites that frequently publish misinformation. To calculate this metric, they daily download the most popular URLs on Facebook and Twitter using NewsWhip, a service that tracks the creation of URLs. NewsWhip also gathers the engagement data, which is an indicator of aggregate engagement (reactions, shares, comments) for Facebook and a "Twitter Influencer Shares" (sum of tweets and retweets of the URL by tracked "influencers") value for Twitter. Every day the 5000 URLs with the highest engagement are selected for both sites. Each URL is then classified based on its rating by NewsGuard or Media Bias/Fact Check (MBFC) as a fallback if no rating is available by NewsGuard. Depending on the rating, the URLs are classified as "Iffy", "OK", or "Unknown" if neither NewsGuard nor MBFC have rated the site (cf. figure 2.14). While these site-level judgments might lead to overestimating the absolute amount of misinformation, the amount of misinformation should be stable and not affect the comparison over time. Finally, they calculate the percentage of URLs from "Iffy" sites and apply a seven-day moving average to smooth out the graph. Values of a day are updated up to 90 days afterward until they are frozen. The visualization shows interesting trends like an increase in the run-up to the 2016 US election and the recent 2020 US election. In between, a clear downwards trend could be observed where Facebook's methods seem to be more efficient, showing a lower "Iffy" Quotient than Twitter since the beginning of 2019 (cf. figure 2.12). Looking at the engagement-weighted graph, though shows that the difference might not be as apparent with Facebook having "Iffy" content with higher estimated engagement from mid-2020 to mid-2021 except for a spike in late 2021 on Twitter which is all very close around the 2020 US election (cf. figure 2.13).

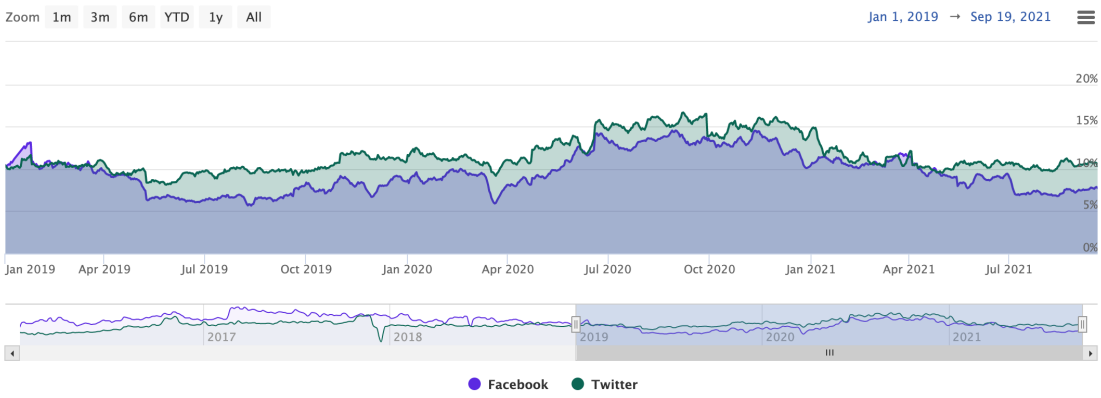

Figure 2.12.: The "Iffy" Quotient from 01.01.2019 to 19.09.2021 (retrieved from[43]).

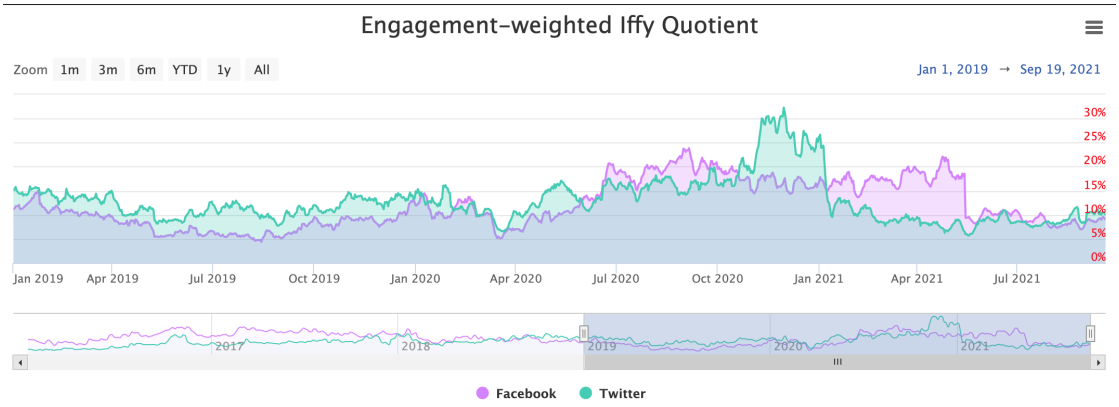

Figure 2.13.: The engagement-weighted "Iffy" Quotient from 01.01.2019 to 19.09.2021 (retrieved from [43]).

This concept would be beneficial not only to big platforms like Facebook and Twitter but also to content providers in general. It would allow them to observe the state of their content and compare themselves to other content providers, which might motivate them to take action. However, this would require an URL-based version of the "Iffy" Quotient since the site-based approach is not fine-grained enough for this purpose.

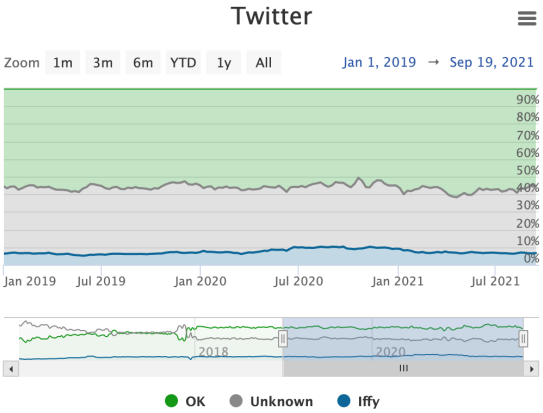

Figure 2.14.: The percentage of "OK", "Unknown", and "Iffy" ratings for Twitter from 01.01.2019 to 19.09.2021 (retrieved from [43]).

2.4.9. NewsGuard

NewsGuard [66] provides a plugin that shows trust ratings for news websites created by trained journalists rather than algorithms. They state that they rate more than 6000 news and information sites responsible for 95% of engagement with news in the US, UK, Germany, and Italy [68]. Each site checked is rated as "Green" if it generally satisfies their criteria, "Red" if it fails to satisfy their criteria, "Satire" if the site is not an actual news website, or "Platform" if the site mainly hosts user-generated content. Nine criteria award a certain number of points, with 100 points being reachable. A site needs at least 60 points to receive a "Green" rating. The criteria are categorized in credibility criteria like "Does not repeatedly publish false information" and transparency criteria like "Website discloses ownership and financing". A complete list of their criteria and rating process rating is available on their website (see [67]).

Besides the rating that is shown next to links in search engines or the browser add-on bar when on a rated site, the premium membership also allows users to see a detailed rating, including a description why this site fails or succeeds in the different rating criteria (cf. figure 2.15).

While targeted at interested readers, this information can also be interesting to content providers who aim to improve their ratings. While creating textual information comparable to that of professional journalists might be an unrealistic task, it gives an excellent example of how the rating of trustworthiness can be done in a way both useful for content providers and consumers alike and should be used as a reference.

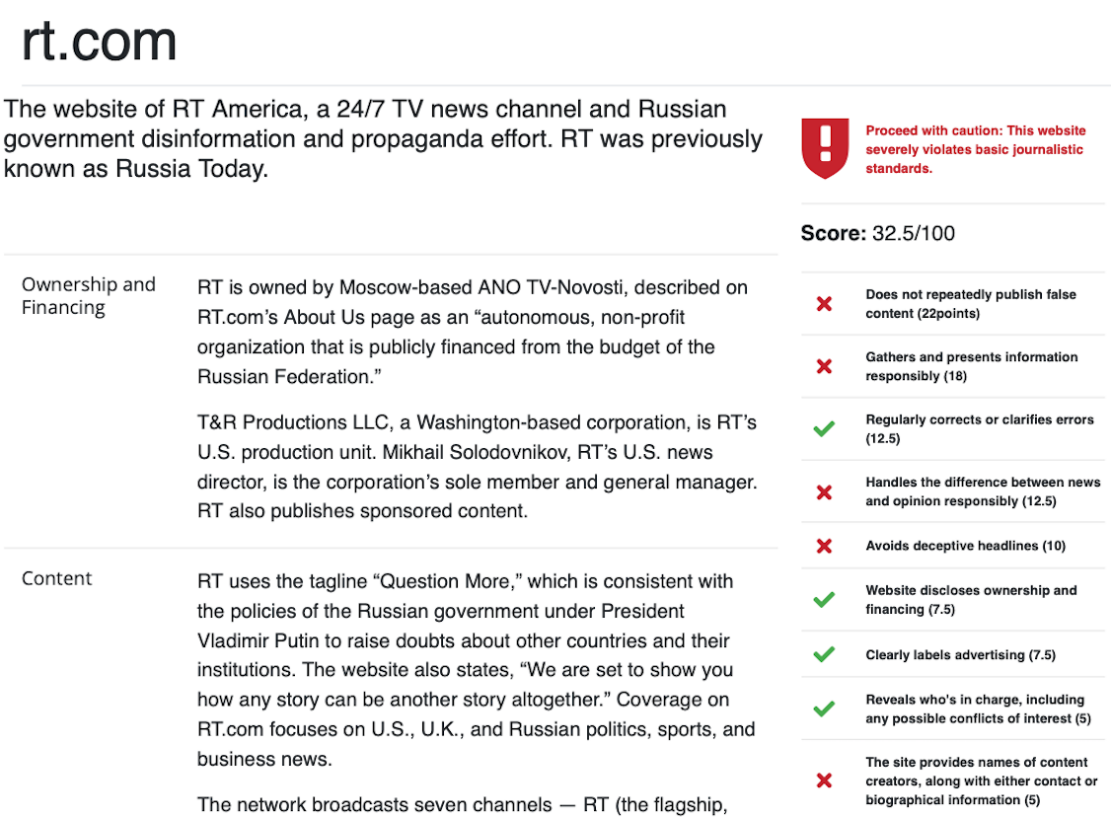

Figure 2.15.: Rating for rt.com by NewsGuard (complete example on `https://www.newsguardtech.com/wp-content/uploads/2020/11/RT-label-2020.pdf`).

2.4.10. BRENDA

BRENDA [9] is a browser extension that supports users to do fact-checking without leaving the website. Other than NewsGuard, the entire process of credibility assessments is automatic. It uses a tested deep neural network architecture to identify check-worthy claims and verify the truthfulness of those claims based on online evidence. Additionally, it also provides evidence snippets to the user. The importance of words and sentences for the classification is highlighted. Users can choose to check only the selected text or the whole article (cf. figure 2.16). The extension also allows the user to give feedback on the results if they think it is inaccurate, which could be used to improve the classifier.

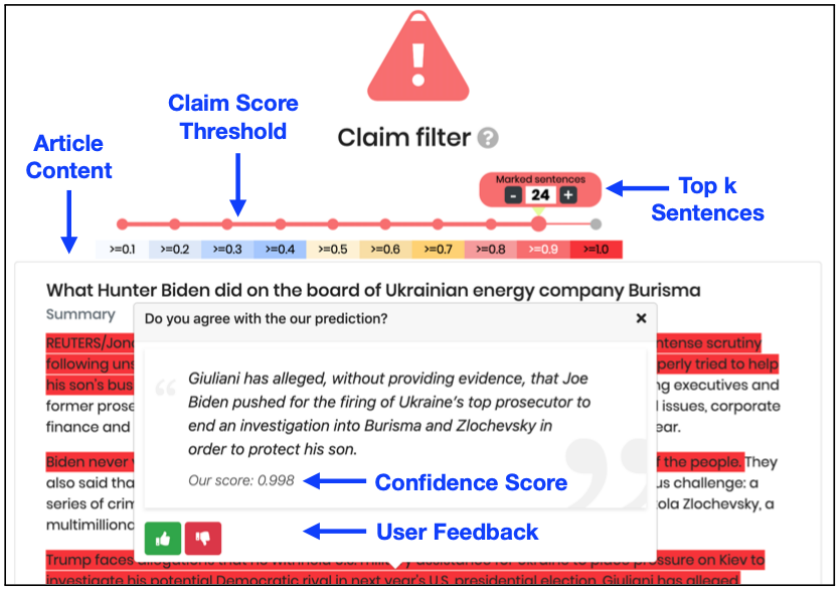

Figure 2.16.: BRENDA analyzing a whole article. The top-scored claim is fact-checked automatically, with the user being able to explore more on-demand [9].

BRENDA shows very well how a browser extension can be done using automatic fact-checking and that user feedback could be important since automatic fact-checking will not have the accuracy of manual fact-checking. Thus, mistakes are likely to happen, but when adding feedback by the user to the model, they could be reduced over time. BRENDA is also another example of highlighting parts of the text that were important for the classification to help interpret results (cf. figure 2.17). The focus of the browser extension is on content consumers. While the functionality could also be helpful to content providers, they would have to manually scan their content without having an overview of their domain.

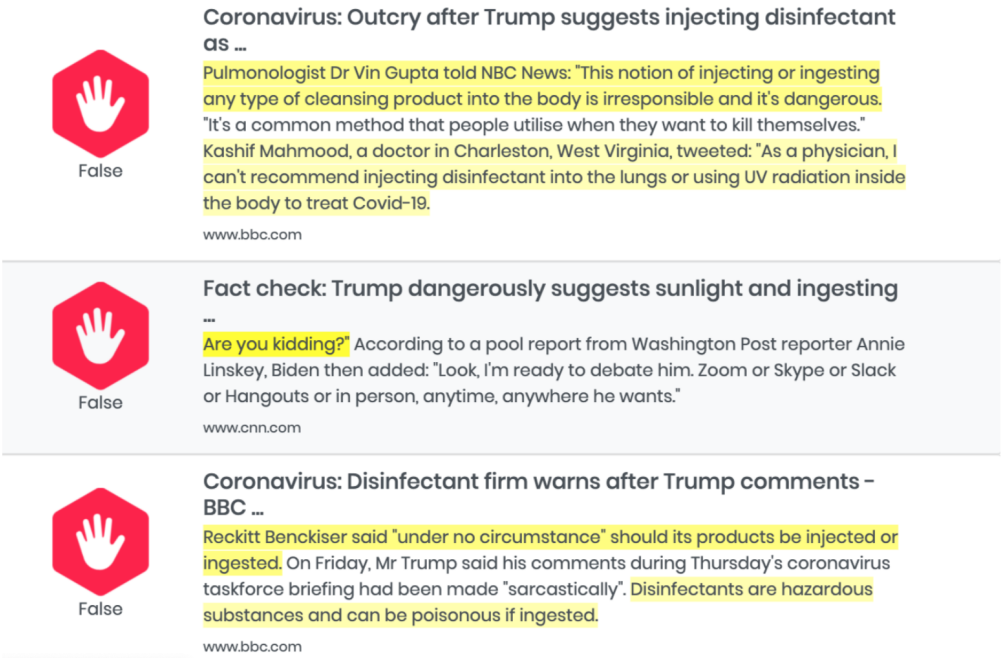

Figure 2.17.: Evidence visualization for the claim "Covid-19 can be cured by ingesting disinfectants" using the attention weights [9].

3. Conceptual Foundation and Design

This chapter presents the concept resulting from the three research questions of this thesis. The first section (3.1) further specifies the research questions into a list of functional and non-functional requirements, which are the basis for both concept and implementation of the prototype. Section 3.2 then lists the different metrics proposed by the thesis to help content providers assess the quality of their content. Next, section 3.3 presents a concept for the regular data collection required to monitor changes over time. Section 3.4 then presents the concept for the dashboard implemented in this thesis. Finally, section 3.5 proposes the concept for collecting user data to analyze and potentially improve the prototype after its release.

3.1. Requirements

This section lists all functional and non-functional requirements to the concepts and the prototype developed in this thesis. Section 3.1.1 contains the functional requirements, conceptionally implemented in sections 3.3 and 3.4 and implemented in chapter 4. Section 3.1.2 contains the non-functional requirements, which will only be relevant for the implementation in chapter 4.

3.1.1. Functional Requirements

- **F1:** A content provider has an overview of all content on a domain to get information about its state. This overview is provided by visualizations of different metrics gained from data about misinformation.
- **F2:** A content provider can see the daily development of the different metrics to identify improvement or deterioration.
- **F3:** A content provider can use the overview to compare their domain to other domains.
- **F4:** A content provider can see detailed information about a single instance to evaluate how they can potentially improve it. It needs to be possible to identify the exact value responsible for any metric's value.
- **F5:** A content provider can search for instances by filtering by name, date, and value of the metrics in the overview and then navigate to the detailed instance view to find problematic (e.g., critical or rumorous) instances.

- **F6:** The service provider (of the dashboard) can see anonymous data regarding the dashboard usage like average page clicks, interactions with the page, independent users, or returning users to evaluate and further improve the dashboard.

3.1.2. Non-functional Requirements

- Performance
  - The delay should be as low as possible. In the best case, it should not be higher than users usually expect from other web pages (for example, between 1-2 seconds and no more than 3 seconds [3]).
  - Unavoidable delays that would lead to a long loading time should be handled asynchronously and communicated to the user (see usability).
- Usability (Some fundamental criteria for usability the prototype should meet)
  - The dashboard should be easy to use, self-explanatory (e.g. providing a help page and/or info tooltips), and follow best practices for visualizations. Fitting visualizations for the data need to be chosen.
  - More complex elements should be explained inside the application (e.g., info button that explains the element on click).
  - Interactive elements should be designed the same way users are familiar with other browser applications (e.g., blue and on hover underlined links).
  - Longer loading times should be done asynchronously and shown with a progress bar.
  - Colors that are used in the visualizations should be customizable for users with color blindness.
- Compatibility
  - The dashboard will primarily be designed for larger screens as mobile devices are not suitable for presenting a large amount of data.
  - The prototype will be accessible via the web browser. Therefore, it is accessible on a wide array of devices, although it will not be optimized for all of them, as mentioned in the previous point.

3.2. Metrics

The first goal of this thesis is to find different metrics that can be useful for content providers to observe the state of their domain and reduce misinformation. Content being straight out incorrect is not the only thing that can cause misinformation. It can also be presented in a misleading way, omit relevant facts, or contain rumors that might turn out to be false later. The metrics presented in this thesis aim to cover multiple different potential sources of misinformation.

As shown by NewsGuards various categories (see [67]), another essential factor is that content providers are transparent and trustworthy. Even correct content might be considered misinformation by users if they do not trust the content provider, which unintentionally leads to misinformation by publishing correct information. Therefore, also metrics that track transparency and trustworthiness should be considered.

The following sections will present various metrics and how they are derived using the data provided by the FactCheck++ framework. The initial description of each metric will be kept generic, so they can be applied using different algorithms or frameworks. Section 3.2.5 will present metrics that might be useful for content providers but can currently not be calculated with the data provided by the FactCheck++ framework.

3.2.1. Instance-level "Iffy" Quotient - Critical Instances

The "Iffy" Quotient presented in section 2.4.8 is a metric describing the amount of "Iffy" content on a social media platform. While it is an excellent measure to estimate "Iffy" content on social media platforms, its site-based definition of "Iffy" is unsuitable for content providers.

However, the same concept can be easily applied by simply changing the definition of what is considered to be "Iffy" depending on the data available by the used algorithm. Best suited for this metric would be a probability of some content being misinformation, like the credibility score of Boididou et al. [6] or the prediction of XFake [73] together with a threshold that decides when content is considered "Iffy".

Nevertheless, even if a prediction does not directly state whether or not content is likely to contain misinformation but only that a particular feature was detected, the metric can still be used to attract attention of a content provider. An example would be ClaimBusters [31] check-worthy claims and also conflicts in the FactCheck++ framework.

In the FactCheck++ framework, "Iffy" instances are referred to as *critical instances*.

Definition (critical instance)**.** *An instance is considered critical if at least one of its facts is critical.*

Simply considering all facts as critical that have at least one conflict with another fact would be too sensitive since conflicts are common. One instance that contains false information would cause all other instances to be marked as critical. A better approach is to use the approval of each fact, which considers both conflicts and matches, to avoid marking facts as critical that have more matches than conflicts. A fact could be considered critical if its approval is lower than a set threshold. However, a significant drawback of the approval is that it weighs all conflicts equally, ignoring that they can have a low approval themselves. The following two examples show two situations where this is problematic:

1. Considering three instances with a fact "rating", which contains a user rating specific to the domain the instance was retrieved from. It can be expected that all values are different, resulting in an approval value of 0. No matter the threshold, the fact would be critical. In the future, the FactCheck++ framework should detect facts that cannot be compared and handle them differently (in Google's rich results, for example, they are not fused), but currently, this issue needs to be circumvented.

2. Considering four instances ("A", "B", "C", "D") with a fact "age", which should be the same for all values. Instances "A" and "B" have the same value, 35, "C" has the value 40, and "D" has the value 45. Instances "A" and "B" would both have an approval value of 0.34, while instances "C" and "D" both have an approval value of 0. Assuming that a fact needs an approval value of at least 0.5 to be considered non-critical, all four facts of the instances are critical. No matter how the threshold is set, a larger number of conflicts will always weigh more than a smaller number of matches, no matter how inconsistent the conflicts are.

The approval value cannot differentiate if two fact values conflicting with the local fact are matching or conflicting with each other. The first is a hint that the local fact value is incorrect, the latter that the fact value should not be compared or was not compared correctly.

A separate rating for matches and conflicts is introduced to overcome this issue:

Definition (match rating)**.** *The match rating is the sum of approval values of all facts considered equal to the fact. It is divided by the sum of all approval values, so the value always lies between 1 and 0.*

Definition (conflict rating)**.** *The conflict rating is the sum of approval values of all facts considered conflicting to the fact. It is divided by the sum of all approval values, so the value always lies between 1 and 0.*

The ratings use the approval of the facts to weigh them differently depending on how conflicting they are. Using the ratings, a *critical fact* can be defined as follows:

Definition (critical fact)**.** *A fact is considered critical if its conflict rating is higher than its match rating.*

This definition changes the assessments from the previous examples.

All facts of the first example would be considered non-critical since they have a match rating and conflict rating of 0. In the example, this assessment is correct since the fact is not comparable. However, it can be argued that those facts should still be critical since it cannot be known with certainty that the conflicts are due to the fact not being comparable. Nevertheless, due to those facts being common in structured data, it was decided that until the FactCheck++ framework can recognize facts that need to be handled separately, it is more beneficial to reduce the number of false critical facts.

The second example results in the instances "A" and "B" being non-critical with a match rating of 1 and a conflict rating of 0, and the instances "C" and "D" being critical with a match rating of 0 and a conflict rating of 1. Adding more conflicting facts would not change this, as long as they disagree.

3.2.2. (Potential) Rumors

A rumor is an information that at its time of creation cannot be verified [79]. Therefore, content providers should avoid rumorous content because of its potential of containing misinformation. The survey of Zubiaga et al. [79] found multiple methods to detect rumors, but they are focused on social media and not on a single domain.

A possible approach could be to detect check-worthy claims, like Hassan et al. [31], attempt to automatically fact-check them, and mark all claims as rumors where the method finds neither proof nor disproof. Content providers can then manually check if they can confirm the validity of those claims, clarify them (e.g., by adding sources or other proof), or remove them.

In the FactCheck++ framework, unique facts are the closest equivalent to rumors. While not being a perfect match, since unique instances might also be the result of content providers assigning values to the wrong fact (e.g. "publishedDate" instead of "datePublished"), from the perspective of the framework those values cannot be confirmed, which matches the definition of a rumor. Furthermore, potential usage of the wrong fact deserves attention as well. An example of a rumor in the FactCheck++ framework could be a "publishedDate" of an entity (movie, book, etc.) where no date was announced yet. Due to not being a perfect match to the definition of rumors, the metric will be referred to as *potential rumor*s in the FactCheck++ framework.

Definition (potential rumor)**.** *A potential rumor is an instance that has at least one unique fact.*

3.2.3. Coverage and Complete Instances

Misinformation is not only caused by false facts but also by omitting parts of the information that is important to understand the topic thoroughly. de Regt et al. [16] identified *"selectively using and omitting facts"* as a strategy used in the health and beauty industry to improve the image despite undesirable scientific discoveries. In politics, there are countless examples of information being omitted and cherry-picked, like the claim that "Brexit" would bring back control over 350 million pounds which completely ignored the UK's budget rebate and consequences by trade disruption [65]. Content providers should therefore try to keep information on their domain complete.

For structured data, it is relatively easy to detect missing information if the fact set of the entity is known. If a fact in this fact set has no value in a particular instance, the instance is missing information. If the correct fact set is unknown, another explanation for missing values can be that another instance assigned values to the wrong fact (e.g., "publishedDate" instead of "datePublished"). Determining the correct fact set of an instance is a future challenge for the FactCheck++ framework and lies outside the scope of this thesis. The *coverage* is used to measure how complete an instance is.

Definition (coverage)**.** *Coverage is the ratio between the local facts of an instance and the number of facts in the fact set of its entity.*

Sometimes it can be more helpful to differentiate if an instance is complete or not, e.g., to see the fraction of complete instances on the domain.

Definition (complete instance)**.** *An instance is considered complete when it has a coverage of 1, i.e., it has a value for every fact of the fact set of its entity.*

3.2.4. Instance Rating - Page Ranking

Content consumers browsing the web might not always be interested in detailed information about how the content they are looking at is rated in different categories. They might just want to know if it can be considered to be trustworthy or not. When there are multiple metrics or categories, they are usually combined to a single rating that allows one to judge whether the content is considered trustworthy (e.g., [66, 9, 6, 73]). Newsguard, for example, sums up points from all its different categories and shows a simple icon to tell users if a domain has passed their criteria [67]. In the area of automatic misinformation detection, Boididou et al. [6] combine the results from various features into a credibility score and label a tweet as fake or real.

The IdaFix component of the FactCheck Framework already has a rating called *Page Ranking*. However, since it was developed previous to this thesis, it does not use all metrics presented here.

Four different metrics are used for the rating, each of them giving points up to a total of 140 points:

- *Equal-to-conflict ratio:* The equal-to-conflict ratio is the fraction of equal fact comparison results to the number of comparisons. Up to 40 points are given depending on the value ($v > 0.2 = 20$, $v > 0.5 = 30$, $v > 0.8 = 40$). Additional 10 points are given if the approval is higher than the average of all instances leading to a maximum of 50 points.
- *Coverage:* The coverage metric presented in section 3.2.3. Up to 40 points are given depending on the value scaling linearly. Additional 5 points are given if the coverage is higher than the average of all instances leading to a maximum of 45 points.
- *Unique facts:* If a unique fact exists, 20 points are given, plus an additional 5 points if more unique facts exist than the average of all instances leading to a maximum of 25 points.
- *Freshness:* The average ratio of remote missing facts to the number of local facts of an instance. If the average freshness is higher than the average of all instances, 20 points are given.

The ranking in the range from A-E is then calculated using both the numerical score and a set of conditions that need to be fulfilled:

- **E:** The lowest rating. None of the conditions are met.
- **D:** A score of at least 50 points is required.
- **C:** A score of more than 50 points is required. Also, the coverage needs to be higher than 0.4, AND the approval needs to be higher than 0.3.
- **B:** A score of more than 70 is required. Also, the coverage needs to be higher than 0.6, OR the approval needs to be higher than 0.6.
- **A:** The highest rating. A score of more than 80 is required. Also, the coverage needs to be higher than 0.6, AND the equal-to-conflict ratio needs to be higher than 0.6.

This rating has three major issues:

1. As described in 3.2.1, the approval has significant drawbacks to determine if a fact is critical since it does not weigh matches and conflicts. The same issue is valid for the equal-to-conflict ratio. The solution was the introduction of a match and conflict rating to define critical facts. They should replace the equal-to-conflict ratio as well.
2. As described in section 3.2.2, unique facts are similar to rumors, which means they can contain misinformation. Despite this, unique facts can give up to 45 of 140 points (including freshness), which contradicts the ambiguity of rumors. Unique facts should at most have a negligible influence on the rating, so inventing facts is not encouraged.
3. Content providers need to have a good understanding of the rating to improve it. However, the rating with both points and conditions is overly complicated. In section 3.4.2, figure 3.13 shows a concept for the visualization of the Page Ranking of single instances. While it shows all information required, it is too complex and requires additional explanation.

Therefore, a different instance rating is proposed. It also uses four metrics which together give up to 110 points:

- *Critical instance:* If the instance does NOT match the definition of a critical instance (see section 3.2.1), it will be given 20 points.
- *Critical fact ratio:* The critical fact ratio is the difference between non-critical and critical facts divided by the number of facts. The result is a value between -1 and 1. Between -40 and 40 points are given depending on the value scaling linearly. The negative points hinder instances with mostly critical facts to get a higher Page Ranking.
- *Coverage:* The coverage metric that was presented in section 3.2.3. Up to 40 points are given depending on the value scaling linearly.

- *Freshness:* The average ratio of remote missing facts to the number of local facts of an instance. 10 points are given if the average freshness is higher than the average of all instances.

Instead of letters, the new rating will use points from 1-5, so it is also possible to calculate average ratings. It is also only based on the points without any other conditions for higher ratings.

1. The lowest rating. The instance has at most 50 points.
2. The instance has more than 50 points.
3. The instance has more than 60 points.
4. The instance has more than 75 points.
5. The highest rating. The instance has more than 90 points.

3.2.5. Other Metrics

- *User rating:* This metric can include ratings retrieved from the domain or ratings collected by the fact-checking system to judge the quality of content. In the FactCheck++ framework, the FeebackBase component is responsible for storing user feedback and using it to influence comparisons. Even if the user rating does not affect the fact-checking system, the information is precious for content providers who should aim for a high user rating.

- *Regularity of misinformation:* Not only the amount of misinformation but also how regular it emerges from a domain can be relevant to content providers. It can be a hint of issues during the creation of content. This metric would require that content is rated as misinformation with relatively high confidence, which the FactCheck++ framework currently does not aim to do.

- *Correction of misinformation:* The main goal of the dashboard presented in this thesis is that potential sources of misinformation are corrected. This metric could measure the progress a content provider makes. If the value is low, content providers should put more effort into correcting content. However, since the FactCheck++ framework currently does not track changes, it is impossible to detect corrections. The equivalent rating of NewsGuard, "Regularly corrects or clarifies errors", also includes that corrections are communicated transparently [67], though this might be difficult to assess automatically.

- *Deceptiveness of headlines and links:* Deceptive headlines like clickbait are sometimes used to increase readership [47]. These headlines are often used to spread misinformation [61] and have shown to correlate to fake news closely [77, 14], so content providers should avoid their headlines being perceived as clickbait. Also, the wording of links leading to content could be similarly misleading and can be included in this metric. FactCheck++ only processes structured data, so headlines are not analyzed. The headline or links to a page, where the structured data was found, could be considered in the future.

- *Trustworthiness of sources:* Sometimes, the sources used to create content might contain misinformation. While this might be done to refute the source, it can still be helpful for content providers to be made aware of potential "Iffy" sources in their content. The FactCheck++ framework currently does not attempt to find sources in structured data, so this metric would not be applicable.

3.3. Regular Data Collection

The functional requirement **F2** is that content providers can see not only the current state of their domain but also the change of the different metrics over time. Therefore, a regular data collection that aggregates the metrics calculated from the instances of each domain is required to fulfill this requirement. If this regular data collection uses all instances on a domain, assuming that the content provider will use the dashboard to improve instances continuously, metrics will start to converge against their optimal value, and new instances will have a decreasing effect. For metrics like critical instances or rumors, a solution could be to show quantities rather than their ratio on the website, but this would not work for metrics like the coverage.

For the "Iffy" Quotient, Resnick et al. [53] use the top 5000 URLs that were published each day on Facebook and Twitter. With this filter, they limit the number of URLs by only choosing those with top engagement and the number of relevant URLs by only choosing those that were published on the given date. Those methods can be used similarly in this thesis. A challenge here is that content providers can differ in the amount of content they produce. A small blog might only publish new content once a week, while a news page can publish numerous content every day. Since content providers should be able to compare multiple domains, it would not work well to let them freely choose the time span.

An alternative could be to limit the number of instances used for the calculation. However, it would be difficult to choose a number that does not result in small content providers effectively having all of their content being considered, while some content of large content providers is never used in the calculation.

The best solution likely is to aggregate the metrics over multiple time spans (1 day, seven days, 30 days, etc.) and let content providers choose from this predefined selection according to their needs. It is a compromise that allows both smaller and larger content providers to use the dashboard while also making regularly collected data comparable.

For simplicity, the prototype developed in this thesis will only use a single time span of 90 days, leaving the implementation multiple time spans as future work.

To differentiate regularly collected data from the most recent data, they will be referred to as *historical data* and *live data*, respectively, in the further course of this thesis.

3.4. Dashboard

The second goal of this thesis is to present the metrics defined in the previous section in an easily comprehensible way for content providers. The dashboard, called Analytics Dashboard, will consist of two separate views to achieve the defined requirements.

The *overview* (section 3.4.1) will allow content providers to see the current state of their domain (**F1**), see the daily development of the different metrics over time (**F2**), compare their own domain to other domains (**F3**), and navigate to the view for a single instance (**F5**).

The *single view* (section 3.4.2) will allow content providers to see detailed information about a single instance (**F4**).

To design the views in a way content providers are familiar with, various dashboards that content providers might be familiar with were examined. The overview concept was inspired mainly by the Google Analytics dashboard [25] and the WordPress dashboard [72]. The single view uses similar elements, while other parts of it are inspired by examples found in related work ([31, 6, 73]) and writing assistants like Grammarly [27].

3.4.1. Overview

Figure 3.1 shows the concept for the overview. It can be separated into three different parts.

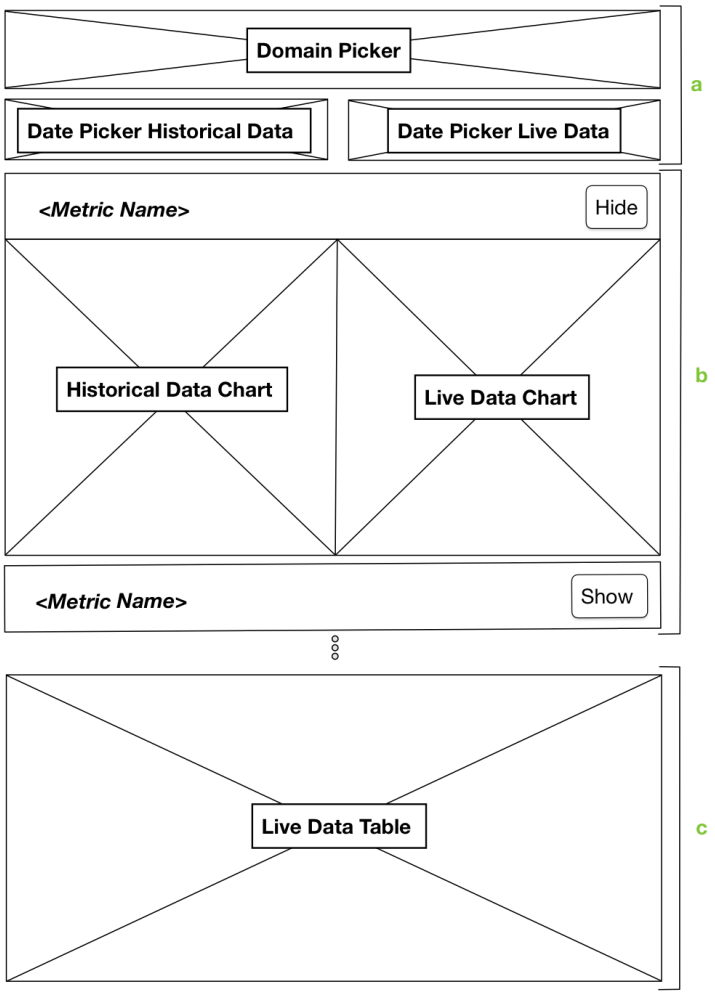

Figure 3.1.: The concept for the overview of the Analytics Dashboard. It consists of 3 parts: (a) the header, (b) the metric charts, and (c) the live data table.

The header (a) contains settings that affect the whole overview.

The *domain picker* shown in figure 3.2 allows users to select the primary domain as well as domains that are observed. The colors used for the domain picker are the same as those used in the charts for the respective domains. Navigating to single instances is only possible for the primary domain, which is the domain owned by the user. A user system that is not part of this thesis can restrict what domains can be selected for the primary domain (or even limit it to a single domain). The domain picker is strongly inspired by the selection of different comparisons in the Google Analytics dashboard [25].

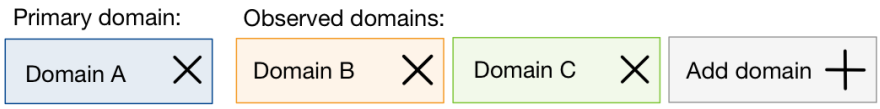

Figure 3.2.: The concept for the domain picker of the overview.

The *date picker* shown in figure 3.3 allows limiting both live and historical data shown in the overview. Users can choose preset dates (b), enter the date manually, or use a date picker to choose (c). It is also possible to select if data should be grouped by days, weeks, months, or years(a). The date picker will change accordingly.

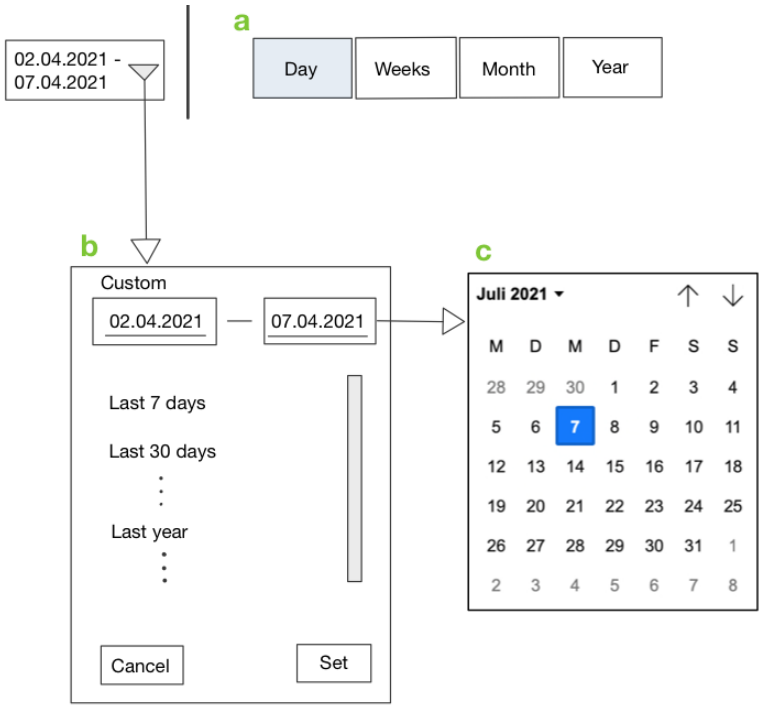

Figure 3.3.: The concept for the date picker of the overview.

In the next part (b), a chart pair for each metric is shown. The user can decide to show or hide the charts for each metric to avoid using too much space and compare charts that otherwise might be too far apart. Both charts show values of the domains selected by the domain picker and are filtered by their respective date picker.

The *historical data chart* shows the historical data from the regular data collection. The user can switch between a line chart (cf. figure 3.4), and a bar chart (cf. figure 3.5). Metrics that can be shown as quotients or quantities also have a setting to switch between those variants.

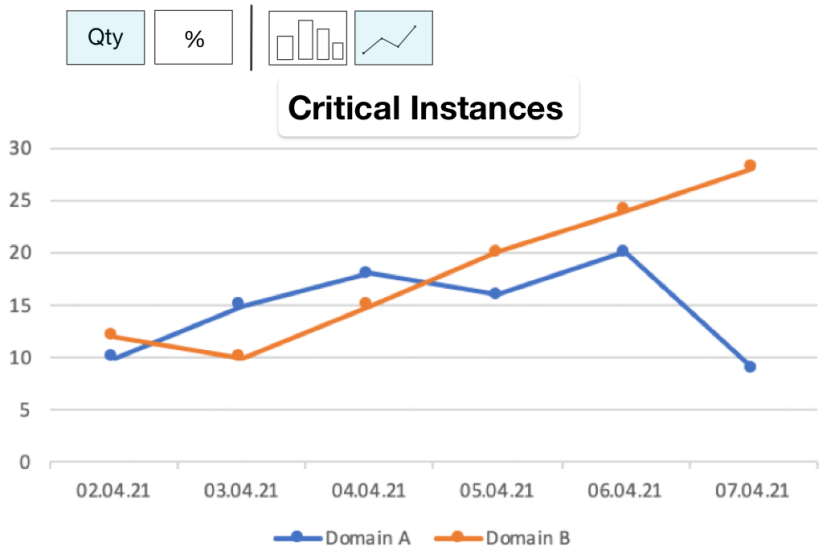

Figure 3.4.: Line chart showing the historical data of critical instances for two domains.

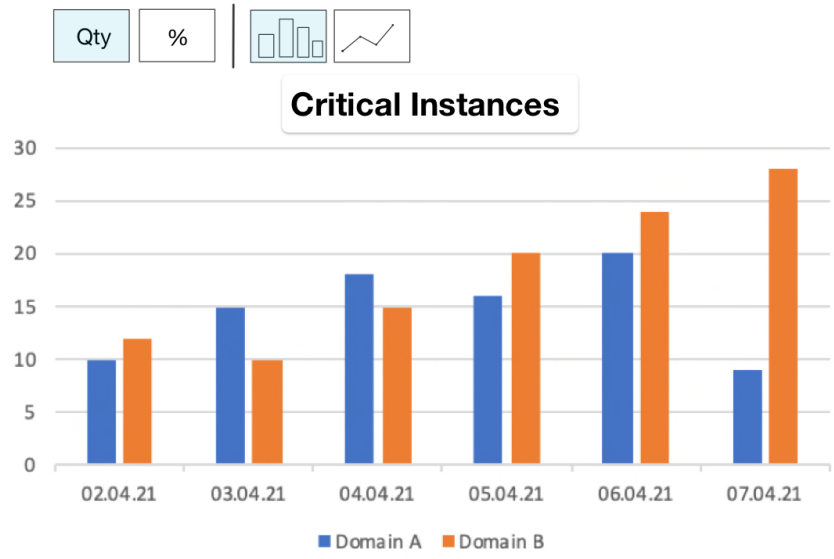

Figure 3.5.: Bar chart showing the historical data of critical instances for two domains.

The *live data chart* (cf. figure 3.6) shows the aggregated metric values of all instances that were created or updated in the time span selected in the live data date picker. The bars are grouped by categories depending on the metric (true and false for boolean metrics, buckets for numeric metrics). The charts can be used to filter the live data table by clicking on the bars of the different categories.

Figure 3.6.: Bar chart showing the live data of critical instances for two domains.

The *live data table* (c) is the connection from the overview to the single view. It shows the name and metric values for each instance of the primary domain, filtered by the live data date picker. Each column is sortable and filterable, either using the live charts or by manually setting filters. On click on a row, the single view of the selected instance will be shown (cf. figure 3.7).

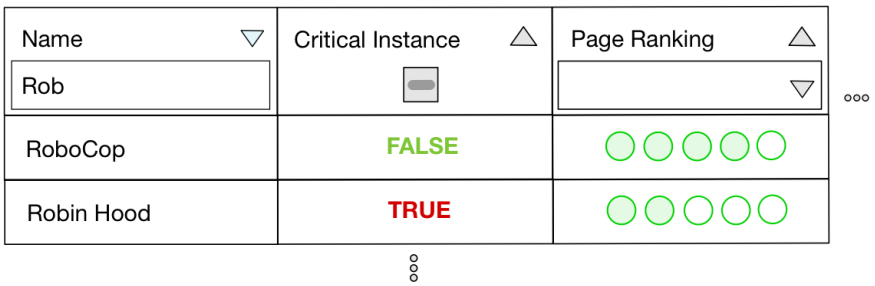

Figure 3.7.: The concept for the live data table of the overview.

3.4.2. Single View

Figure 3.8 shows the the different diagrams of the single view.

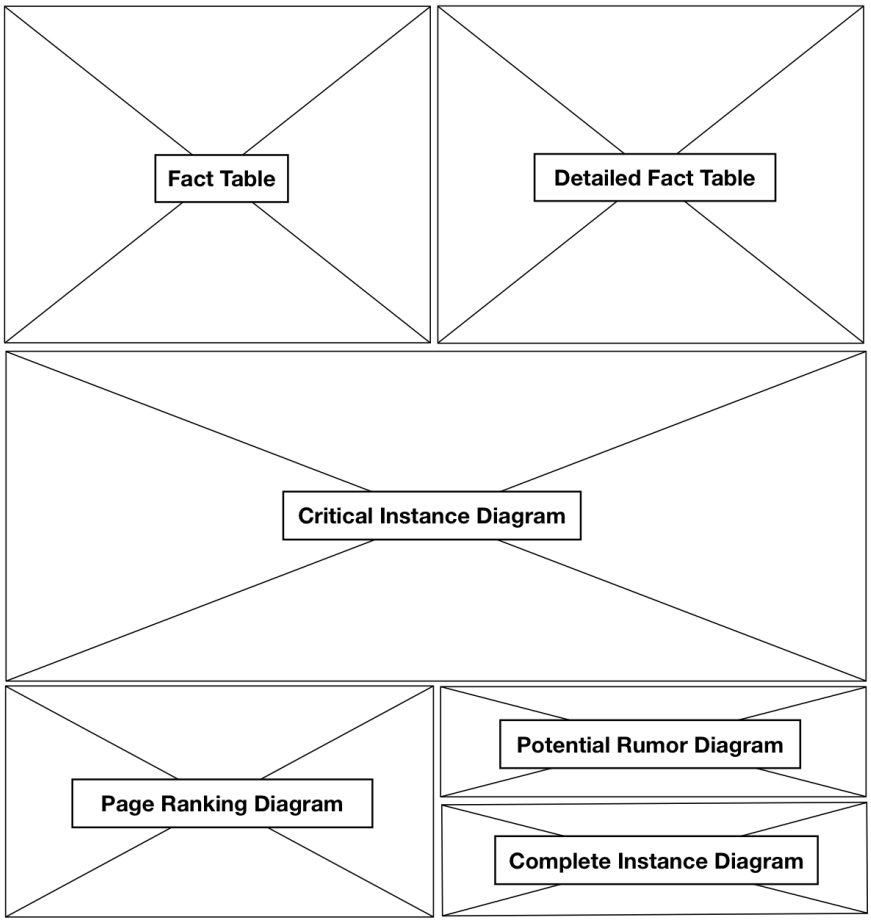

Figure 3.8.: The concept for the single view of the Analytics Dashboard.

The goal of the single view is to allow content providers to identify the cause for all metric values. In the case of the FactCheck++ framework, the origin for all metrics stems from the comparison of fact values. Therefore, showing those values and the comparison results is the most important task of this view. The *fact table* shown in figure 3.9 fulfills this task. All local facts of the instance are shown in a table, followed by the locally missing facts. Fact values are colored depending on their comparison result. The color for each result is shown in the legend on top of the table. The visualization takes inspiration from the highlighting of text as done in related work like ClaimBuster (cf. figure 2.6), XFake (cf. figure 2.8), or BRENDA (cf. figure 2.17), as well as writing assistants like Grammarly [27] that use different coloring to draw attention to different mistakes and popups to show potential fixes.

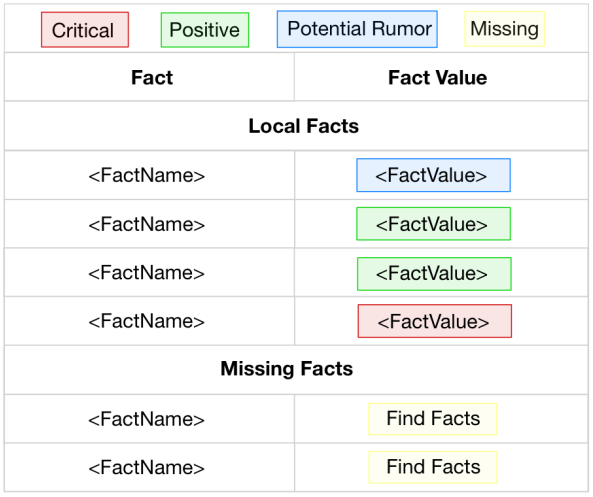

Figure 3.9.: The concept for the fact table of the single view.

At a click on a fact value, the *detailed fact table* for the selected fact is shown (cf. figure 3.10). It shows the Page Ranking, the fact value, and the approval value of the fact from all known instances. The color of the approval value additionally signals if the instance containing the value is critical. The rating area marks a place where additional widgets could be placed that allow content providers to rate the comparison in future iterations. This visualization should support content providers to correct fact values of critical facts. It allows them to check the values of conflicting facts and evaluate if they are a good fit by checking the approval and Page Ranking (e.g., choosing a value with high approval from an instance with a high Page Ranking). For locally missing values, the visualization looks the same except that it shows no local value.

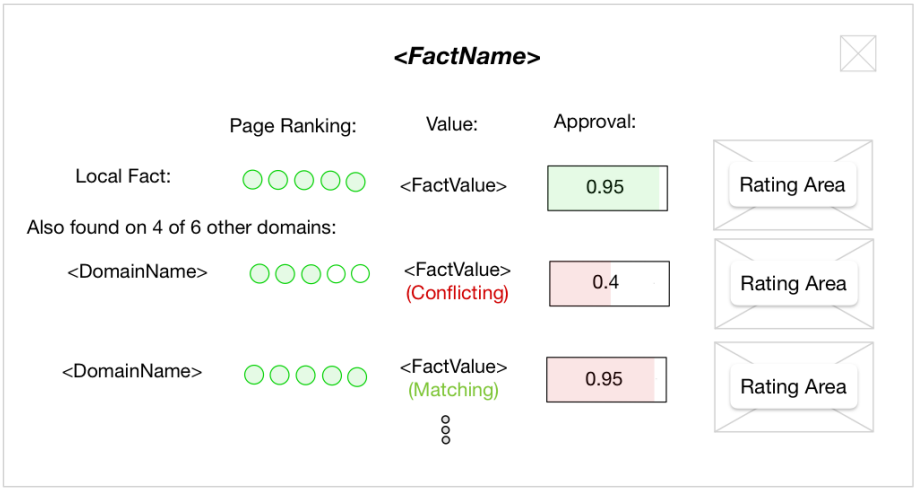

Figure 3.10.: The concept for the detailed fact table of the single view.

While the tables show if a fact is critical or not, it does not show the rating that led to this judgment. The *critical instance diagram* (cf. figure 3.11) shows if the instance was rated critical as well as the match rating and conflict rating for each fact. Three different states of a fact can be seen:

1. Fact A has a higher match rating than conflict rating. Therefore, it is rated as non-critical/positive.
2. Fact B has both a match rating and conflict rating of 0. None of the instances agree so a judgment is not possible. The fact is rated as non-critical/positive.
3. Fact C has a higher conflict rating than match rating. It is rated as critical, causing the whole instance to be rated critical as well.

Like the fact table, this diagram is also connected to the detailed fact table, i.e., a click on the bars shows detailed information for the fact.

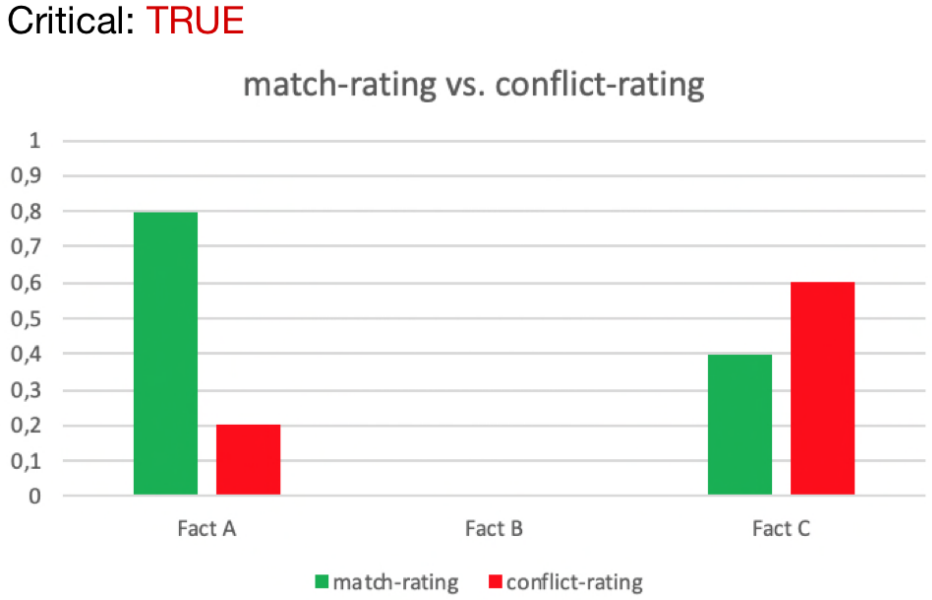

Figure 3.11.: The concept for the critical instance diagram of the single view.

Potential rumors and complete instances are relatively easy to visualize since the comparison result requires no further explanation. Figure 3.12 shows how both of them can be displayed using a progress bar. In the *potential rumor diagram*, the freshness additionally shows how unique the instance is. Coloring the boolean value in the same colors as critical instances (true -> red, false -> red) is avoided to circumvent judging if unique facts are positive or negative. For the *complete instances diagram*, the coverage shows how complete the instance is. Coloring the boolean value in the same colors as critical instances is no issue here.

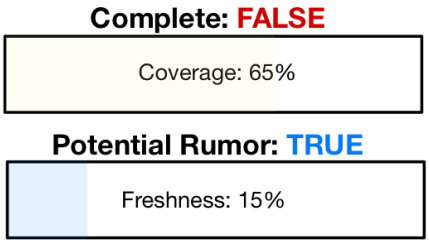

Figure 3.12.: The concept for the visualization of complete instances and potential rumors in the single view.

Figure 3.13 shows the concept for the initial *Page Ranking diagram*, one of the reasons why it was argued to change it. The blue lines mark the average which needs to be exceeded to get additional points, the grey lines mark the thresholds required to reach a rating, and the black lines for the equal-to-conflict ratio mark the thresholds for points since they do not scale linearly. Points for unique facts are given if at least one exists, so it is displayed only with a boolean in the concept.

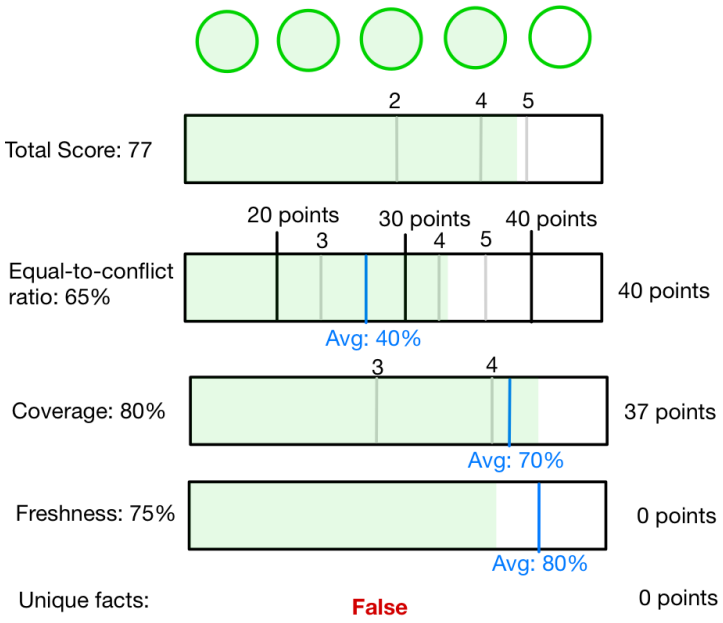

Figure 3.13.: The concept for the visualization of the Page Ranking used in IdaFix.

The changes in the Page Ranking metric simplify the diagram, as can be seen in figure 3.14. It shows an instance with four facts, of which one is critical, making the instance critical as well. The instance also has one missing fact and a freshness lower than the average. With a total of 62 points, this results in the instance having a Page Ranking of 3.

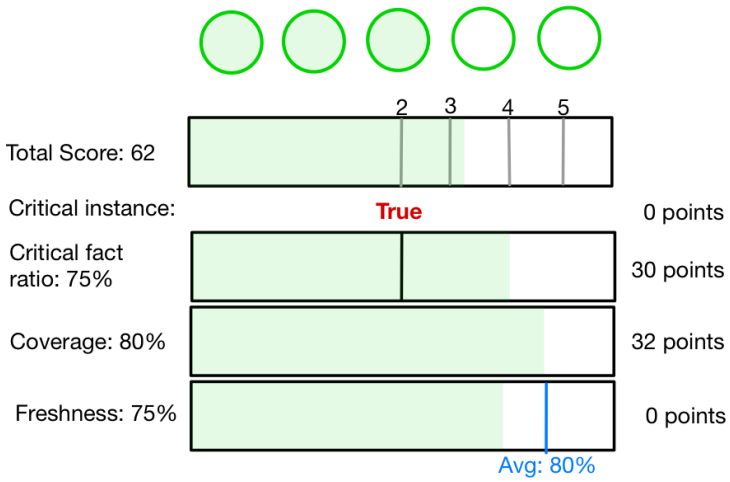

Figure 3.14.: The concept for the visualization of the updated Page Ranking for the single view.

3.4.3. Common

Some parts of the dashboard require a textual explanation. In addition to a dedicated help page explaining all essential terms, info buttons should be placed where terms might be unclear. On click a popup appears with the most important information (cf. figure 3.15).

Figure 3.15.: The concept for the info buttons in the dashboard.

As is standard in most dashboards, all charts that show multiple values should show a tooltip on hovering over it. The tooltip should include the label of the x-axis and the values for each subgroup (e.g., domain) on the y-axis (cf. figure 3.16).

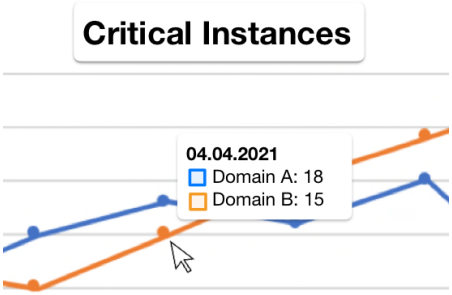

Figure 3.16.: The concept for the hover tooltips in charts of the dashboard.

3.5. User Analysis

The last requirement, the ability of the service provider to see the dashboard usage statistics (**F6**), was not addressed yet.

For the prototype, Google Analytics will be used to fulfill this requirement. Besides automatically collected events like "page_view" when a page is loaded or "scroll" when the user scrolls to the bottom of the page, it is also possible to add custom events to track button clicks or other interactions. The events also yield information about the page URL and page title, so adding the entity's name to the title of the single view could automatically provide information about what entities are checked most often. More information about events can be found at [24].

4. Implementation

This chapter presents the prototypical implementation of the concept presented in chapter 3. The first part of this chapter will focus on the changes required to already existing components of the FactCheck++ framework to implement the concepts for metrics and regular data collection presented in section 3.2 and section 3.3, respectively. Section 4.1 will describe the changes in the database for instances and statistics in the FactBase component. Section 4.2 describes the FactServer, which is responsible for comparisons, calculating statistics, and provides APIs to access this data stored in the FactBase. The calculation of the new metrics and the regular data collection will be added to this component.

Section 4.3 will then present the Analytics Dashboard prototype, built on top of those APIs, that realizes the dashboard concept from section 3.4.

4.1. FactBase

The FactBase uses the NoSQL database *Apache CouchDB* (version 3.1.1) [22] to store instances and statistics. Data in CouchDB is stored using uniquely named, semi-structured JSON documents and accessed via a RESTful HTTP API. The storage in semi-structured JSON documents simplifies saving instances since the properties of structured data can vary widely, so it would be difficult to fit into a relational database. *CouchDB views* bring structure to the stored data and are a replacement for SQL queries. They operate on the documents existing in the database without modifying the original document using a map-reduce system.

4.1.1. Database

Before changes in the Analytics Dashboard were made, instances and statistics were saved in separate databases. They can be described with the ER-diagram shown in figure 4.1 (a), with each database being represented as a separate entity.

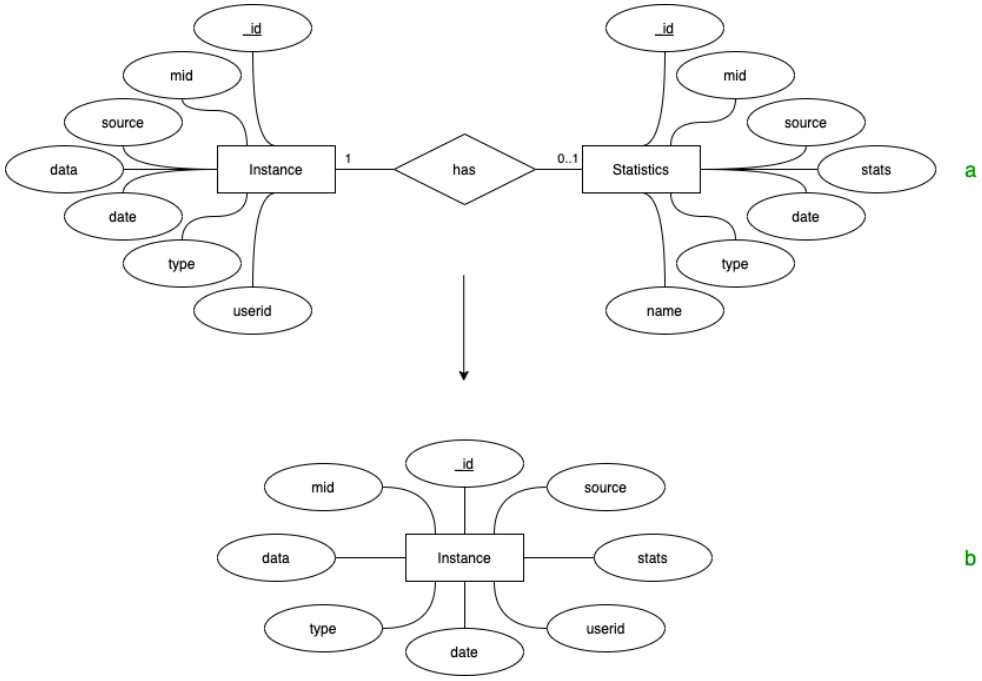

Figure 4.1.: ER-diagram showing the two separate instance and statistic databases as entities (a) and how they were joined into a single database (b).

The attributes contain the following information:

- *_id:* The unique id of the document. CouchDB automatically generates it for every instance. For the statistics, the attribute is set to the identical value serving a similar purpose as a foreign key.
- *mid:* The mid is the id by which instances belonging to the same entity can be identified. It is determined for each new instance by the Resolver of the FactServer, which uses the Google Knowledge Graph API [26] to find a fitting mid.
- *source:* The source of the instance.
- *data:* The JSON object containing the structured data (i.e., fact key-value pairs) retrieved from the source.
- *name:* The name of the instance. The name is part of the JSON object in the data attribute. Therefore this attribute is not required for instances.
- *stats:* The JSON object containing all statistics about the instance. The object will include the new metrics additional to other values which are not relevant to this thesis.
- *date:* The date when the instance was last updated in the database (ISO date-time format).

- *type:* The type of the instance (e.g., movie, person, etc.).
- *userid:* The id of the user that updated the instance.

While this separation was working for the previous dashboard, which only showed statistics with no connection to the facts of the instance, it is inefficient for the Analytics Dashboard, which requires both for the single view. The logic to join instances and statistics would have to be added outside of CouchDB since it considers the databases unrelated. Also, the implicit 1 to 0..1 relation from instances to statistics does not reflect reality since there should always be statistics available even if no comparison has happened yet.

The two databases were therefore merged by adding the stats attribute to documents of the instance entity as shown in figure 4.1 (b). If documents in the old format are required, CouchDB views can filter the data or stats attribute. The database will be referred to as *instance database* in the further course of this thesis.

For the historical data, a separate database was created since it has no relation to the instances in the instance database (cf. figure 4.2).

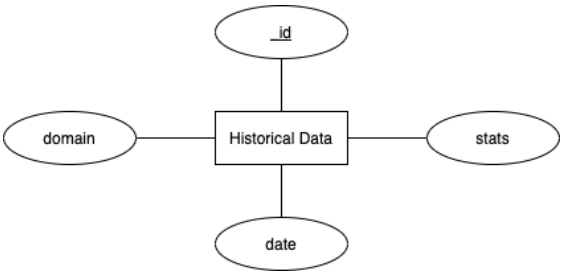

Figure 4.2.: ER-diagram showing the database for historical data.

The document in this database has the following attributes:

- *_id:* The unique id of the document. CouchDB automatically generates it for every new document.
- *domain:* The domain of the instances that were used for the calculation.
- *date:* The date on which the historical data was calculated (ISO date).
- *stats:* The historical data that was calculated.

The database will be referred to as *historical data database* in the further course of this thesis.

4.1.2. CouchDB Views

A CouchDB view was added for each of the two databases to filter the documents by date and domain. The views map the stats attribute as value to its domain and date attribute as key, enabling queries for documents from a set of domains in a given period using the "startkey" and "endkey" query parameters provided by CouchDB. The view created for the instance database also adds the instances name and source to the value since it filters the data attribute, making them no longer available otherwise. It also has a custom reduce method that aggregates all values matching a given filter on demand. Therefore, this method can be used for the regular data collection described in section 3.3 by filtering a specific domain and start and stop dates corresponding to the desired period of time. The view for the instance database will be referred to as *instance view* in the further course of this thesis. The view for the historical data database will be referred to as *historical data view* in the further course of this thesis.

4.2. FactServer

The FactServer is the central component of the framework. It is implemented as Tornado Web Server (version 6.1), a python web framework and asynchronous networking library [4]. It provides a RESTful API to trigger comparisons and retrieve information about instances, instance statistics, and precision metrics. It is also responsible for resolving what entity an instance belongs to, comparing it with all existing instances, and calculating and updating all statistics resulting from the comparison.

Since the FactServer is already responsible for calculating some statistics about the instances, it should also handle the calculation of the new metrics, the regular data collection, and add APIs to access them.

4.2.1. Calculation of Metrics

The calculation of statistics, including the new metrics, starts with a request to compare an instance. Table 4.1 shows a description of the API endpoint that is used to request a comparison.

Title	**Instance comparison**
URL	/get
Method	POST

Parameters	{
	#structured data
	"data": object,
	#source of the instance
	"source": string,
	#id for authorization
	"userid": string
	}

Success	200: comparison results
Error	- 400: Parameters incorrect. - 401: Unauthorized. - 422: Name property missing in data. - 422: Could not resolve entity type. - 422: No other equal instances found.

Table 4.1.: Description of the FactServers Comparison API endpoint.

A shortened example for a comparison result can be found in Appendix section A.3.1. It contains the comparison results for every fact grouped by the instance they were compared with. The same information is used to calculate the statistics. Currently, this endpoint is used by the IdaFix Browser plugin, which extracts the structured data from websites it visits. The user id is used to authorize the comparison and track changes in the instance database. The "422 - Unprocessable Entity" error is returned if the structured data contains no name or if the FactServer does not know the type of the instance. The error also occurs if no other instance of the same entity is known to the FactServer. In this case, no comparison is possible, and both the instance and its facts are unique.

Initially, this request only compared the new (or updated) instance and updated its statistics but did not update the statistics of any other instance. This was changed so that statistics are always correct without requiring additional calls to the API to keep information in the instance database consistent. To avoid documents being updated simultaneously, potentially causing inconsistencies in the instance database, a value-based thread lock is used to allow only one active request per entity. Only the initial comparison required to respond to the request is calculated before any lock is acquired to avoid delays in the response. Figure 4.3 shows a successful sequence of tasks triggered by a comparison request.

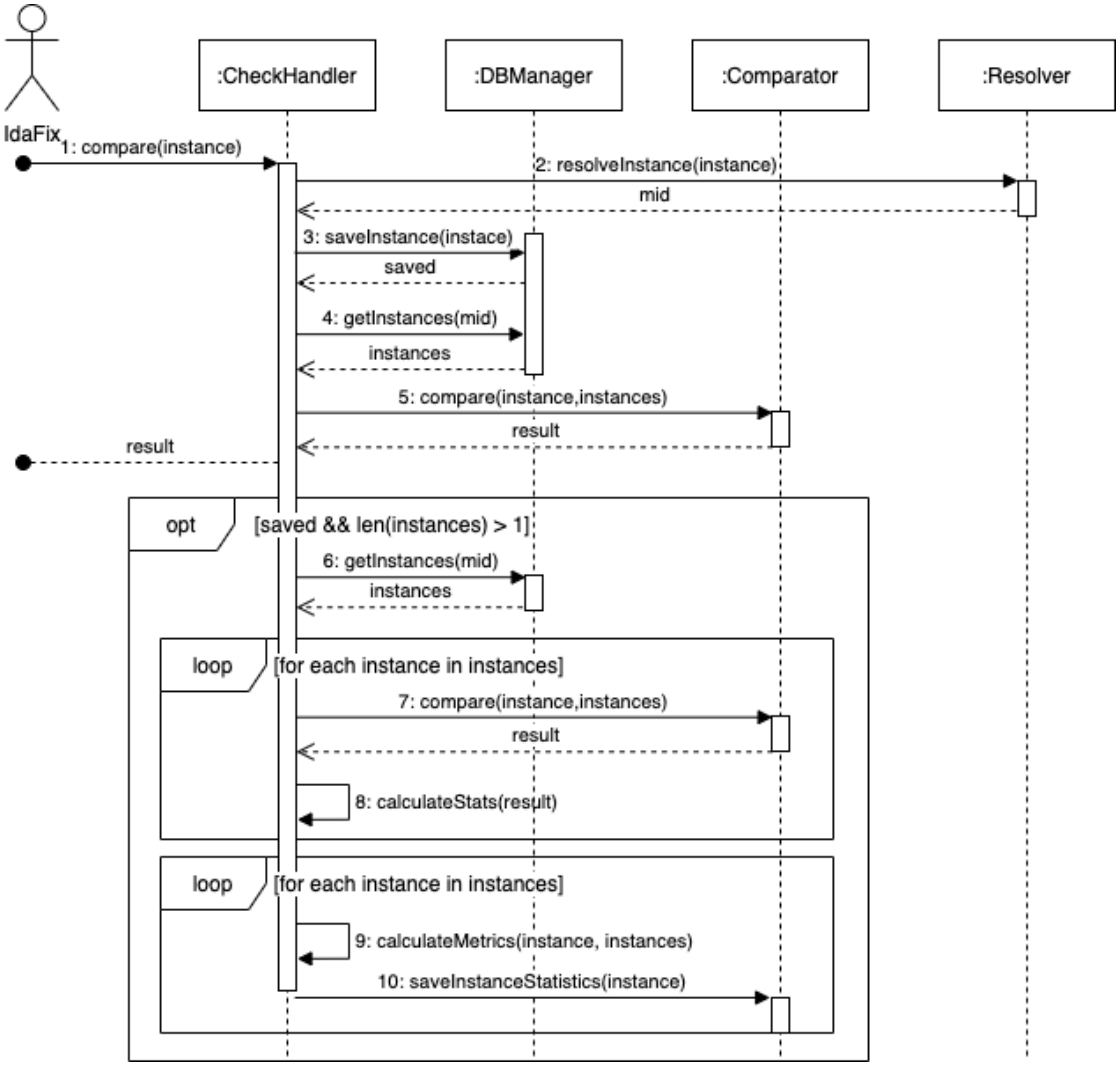

Figure 4.3.: Diagram showing the successful sequence of tasks triggered by a comparison request.

1. *compare(instance):* A user visits a website containing structured data with the IdaFix browser plugin active. IdaFix extracts the data and requests a comparison from the FactServer via the endpoint shown in table 4.1. The CheckHandler checks the request for validity and transforms it into an internal representation of an instance (including default statistics).

2. *resolveInstance(instance):* The CheckHandler requests the Resolver to assign the instance a mid, which is used to identify which entity the instance belongs to. The Resolver will first check if an instance with the same name and source is already known and returns its mid if this is the case. Otherwise, the instance's name will be used to find a name via the Google Knowledge Graph API [26].

3. *saveInstance(instance):* The CheckHandler requests the DBManager to save the instance to the instance database. The DBManager will check if the instance already exists in the instance database and, if it does, whether or not it differs in any way. Only if the instance is new or different, the DBManager will save it and return true to the CheckHandler.

4. *getInstances(mid):* Independent of whether or not the instance was saved, the CheckHandler still has to return a comparison result for the request. For the calculation, the CheckHandler requests all instances of the same entity from the DBManager. The mid is used to identify all instances belonging to the same entity.

5. *compare(instance, instances):* With all instances retrieved, the CheckHandler requests the Comparator to compare the instance to all other instances of the same entity. The result is the comparison results for every fact grouped by the instance they were compared with. This result is sent as a response. All further steps would only delay the response and are therefore processed on a separate thread.

6. *getInstances(mid)*: The following steps are only executed if the instance was saved in step (3) and if there are multiple instances of an entity. Otherwise, recalculating the statistics is not necessary. A value-based thread lock prevents that the following steps happen at the same time for the same entity. Since instances may have changed in the meanwhile, the CheckHandler will request them as described in step (4).

7. *compare(instance, instances):* The comparison described in step (5) is now repeated for each instance retrieved in step (6).

8. *calulateStats(result):* For each comparison result, the CheckHandler will calculate different statistics, including most of the new metrics. The metrics critical instance and Page Ranking cannot be calculated during this step since they rely on other instances' results.

9. *calculateMetrics(instances):* After the calculation of all statistics, the CheckHandler will loop over all instances again and calculate the metrics that could not be calculated in step (8).

10. *saveInstanceStatistics(instance)*: In the last step, the statistics are saved in the instance database by the DBManager.

4.2.2. Regular Data Collection

As mentioned in section 4.1.2, the CouchDB instance view can aggregate all instance statistics of a single domain in a time span, fulfilling the requirements for the regular data collection described in section 3.3. Therefore, the FactServer only needs to regularly request this information for the last 90 days for every domain from the instance database and save it into the historical data database. In Tornado, a regular occurring job can be scheduled using the "TornadoScheduler" provided by the Advanced Python Scheduler library [29] (version 3.7.0). The trigger type "cron" is used to schedule a job for a specific time chosen to be 00:00:00 UTC. A solution for multiple time zones is not planned for the prototype. The job is described by the sequence diagram in figure 4.4.

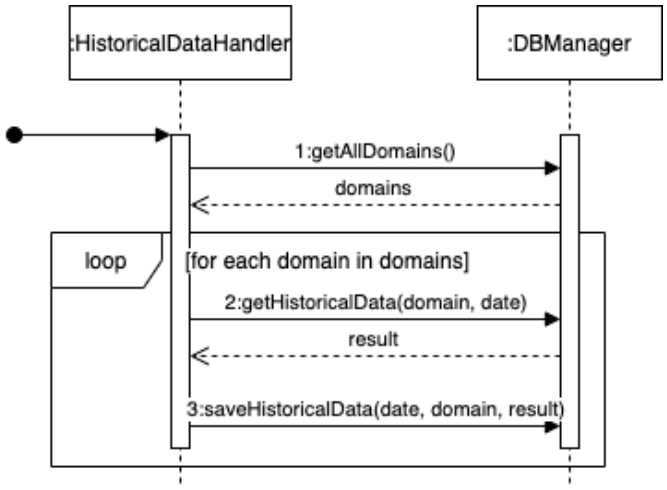

Figure 4.4.: Diagram showing the sequence of tasks for the regular data collection.

1. *getAllDomains():* When triggered at 00:00:00 UTC, a list of all known domains is retrieved from the DBManager.

2. *getHistoricalData(domain, date):* The historical data for the previous day is requested for every domain.

3. *saveHistoricalData(date, domain, result):* The historical data is saved together with the domain and date.

4.2.3. Retrieving Instance Statistics (APIs)

Three new API endpoints were added to the FactServer to retrieve instance statistics.

The first endpoint shown in table 4.2 can be used to retrieve the live data for one or multiple domains in a given time span.

Title	**Get live data**
URL	/stats
Method	GET
Parameters	- domains: string[] of domain names. - start_date: date string in ISO date-time format (optional). - end_date: date string in ISO date-time format (optional).

Success	{
	#domain name from "domains"
	"www.domainA.com": [
	{
	#id in the instance database
	"id": string,
	"name": string,
	"source": string,
	#ISO date-time
	"last_updated": string,
	"stats": {
	"potential_rumor": boolean,
	"freshness": float,
	"loc_coverage": float,
	"complete_instance": boolean,
	"critical_instance": boolean,
	"page_ranking": int (1-5),
	#other statistics
	...
	}
	},
	#more instances from www.domainA.com
	...
	],
	#other requested domains from "domains"
	"www.domainB.com": [...],
	...
	}

Error	400: Parameters incorrect.

Table 4.2.: Description of the FactServers Live Data API endpoint.

The handler for this endpoint uses the instance view without requesting to aggregate the result. The date of the last update ("last_updated") is added to the returned data to allow sorting or further filtering by applications using the endpoint. If no instances can be found or the domain does not exist, an empty list will be returned. A shortened example of the success response can be found in Appendix section A.3.2.

The second API endpoint shown in table 4.3 can be used to retrieve the historical data for one or multiple domains in a given time span.

Title	**Get historical data**
URL	/historical
Method	GET
Parameters	- domains: string[] of domain names. - start_date: date string in ISO date format (optional). - end_date: date string in ISO date format (optional).
Success	{ "2021-10-10": { #domain name from "domains" "www.domainA.com": { "num_potential_rumor": float, "freshness": float, "loc_coverage": float, "num_complete_instances": float, "num_critical_instances": float, "page_ranking": float (1f-5f), #other statistics ... }, #other requested domains from "domains" "www.domainB.com": {...}, ... }, #historical data of other dates "2021-10-11": {...}, ... }
Error	400: Parameters incorrect.

Table 4.3.: Description of the FactServers Historical Data API endpoint (GET).

The handler uses the CouchDB historical data view to retrieve the requested data. The results are then grouped by date rather than domains, as was the case for live data. If no historical data was found for the request, the response is an empty JSON object. A shortened example of the success response can be found in Appendix section A.3.3.

This endpoint also accepts POST requests to update historical data as shown in table 4.4.

Title	**Update historical data**
URL	/historical
Method	POST
Parameters	{ "domains": string[], #ISO date-time (optional) # by default the current day. "date": string }
Success	200: OK
Error	400: Parameters incorrect.

Table 4.4.: Description of the FactServers Historical Data API endpoint (POST).

This call triggers step (2) and (3) shown in figure 4.4 for the chosen domains. The endpoint can be used to calculate historical data although the day did not pass yet or recalculate data of passed days. The latter might result in inaccurate results since instances might have been updated and are no longer included in the calculation, although they would have been in the past. Currently, the POST request to this endpoint is only used for debugging.

The last API endpoint can be used to get detailed information about a single instance. The handler for this endpoint does not only return the data available about the instance but also adds the information from other instances of the same entity. Figure 4.5 shows the activities to create the response for this endpoint. First, the requested instance is retrieved to check if it exists. The statistics about the instance and the facts are then added to the response. Next, all other instances of the entity are retrieved, and their instance statistics are added to the corresponding "other_xxx" object. Then for each fact, it is checked if it is missing, conflicting, or equal. Depending on the result, the statistics about the fact are added to "missing_fact_stats", or to the corresponding fact in "co_facts", or "eq_facts".

Table 4.5 shows the description of this endpoint, including the response. A shortened example of the success response can be found in Appendix section A.3.4.

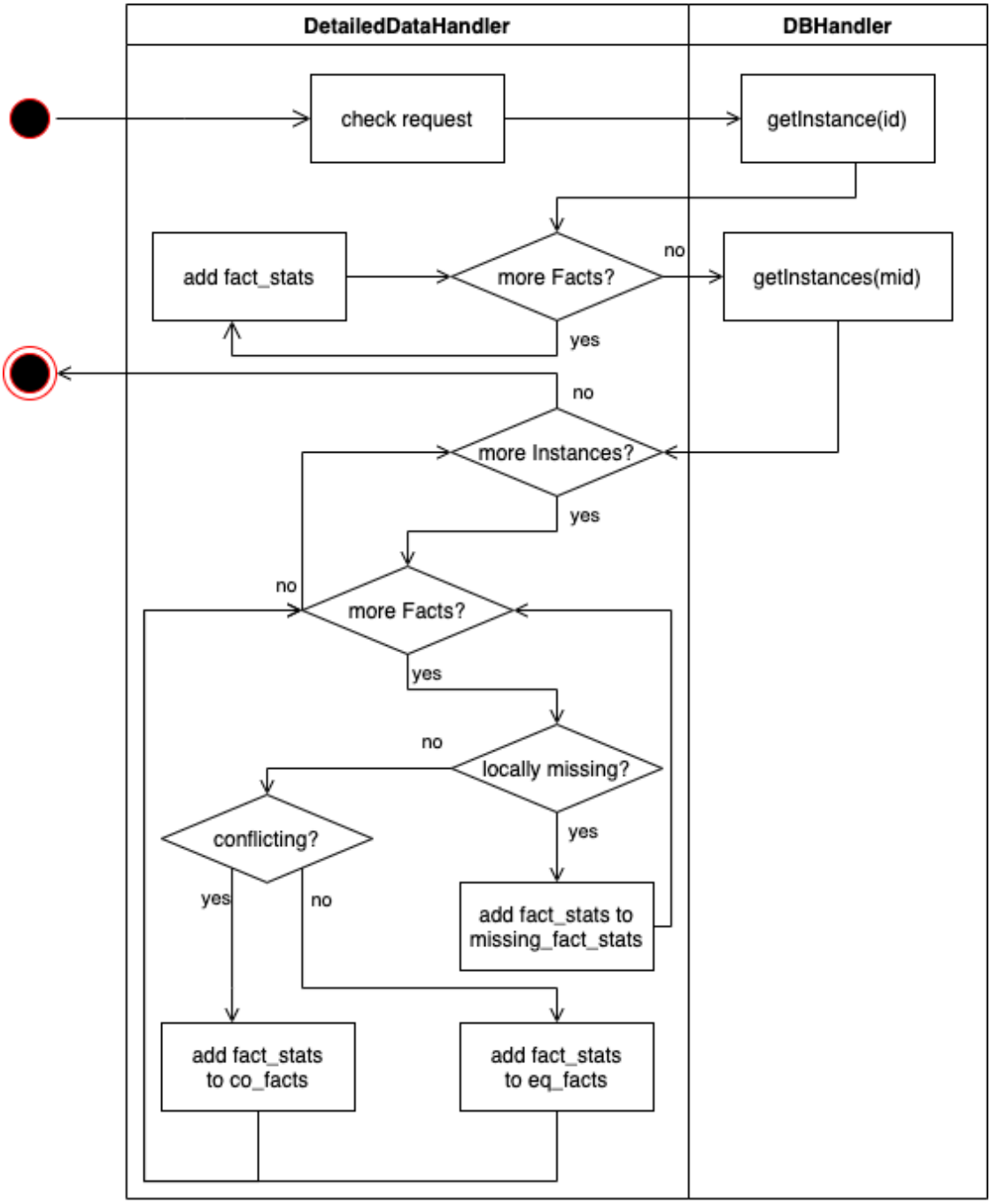

Figure 4.5.: Diagram showing the activities to create a detailed data response

Title	**Get detailed data**
URL	/stats/{id}
Method	GET
Success	{ "source": string, "type": string, "potential_rumor": boolean, "page_ranking": float (1f–5f), #other statistics ... #statistics for each local fact "fact_stats": { "name": { "approval": float, "unique_fact": boolean, "match_rating": float, "conflict_rating": float, "critical_fact": boolean, #the local fact value "value": any, #fact_stats for equal facts "eq_facts": { "www.domainB.com": { "approval": float, "unique_fact": boolean, "match_rating": float, "conflict_rating": float, "critical_fact": boolean, #the remote fact value "value": any }, #other domains ... }, #fact_stats for conflicting facts "co_facts": { #see eq_facts } }, #statistics of all other facts ... }, "missing_fact_stats": { #see eq_facts }, "critical_instance_other": { "www.domainB.com": boolean, #values for other domains }, #other statistics of other domains }
Error	404: instance with id not found.

Table 4.5.: Description of the FactServers Detailed Data API endpoint.

4.3. Analytics Dashboard

The Analytics Dashboard is the prototypical implementation of the concept presented in section 3.4. It was implemented using SvelteKit (version 1.0.0-next.151) [64], a framework to build web apps with best practices like build optimization, prefetching, and configurable rendering (server or client-side) already realized.

A combination of Svelte and D3.js [8] was used to generate SVG charts. While it turned out that generating and animating the SVG is easier using loops and animation directives in Svelte, the D3.js library was still valuable for its functions to create scales mapping values to positions in the SVG or to generate axes for the chart. For the style, a customized Bootstrap [7] style sheet was used (version 5.1.0).

A selected set of structured data from non-existing domains were added to the instance database to present how the concepts were implemented. The historical data in the historic data database was also modified to show a more realistic picture of a domain with constant changes. Since state and modification of actual data cannot be controlled, it would not be suitable for this purpose. A list of the structured data used can be found in Appendix section A.4. The prototype can be accessed from the network of the University of Vienna through the following link: `http://fcheck02.cs.univie.ac.at:3000`.

4.3.1. Overview

Figure 4.6 shows the implementation of the overview.

As seen in the domain picker at the top, "www.movieFacts.at" is selected as the primary domain, and "www.rottenmelons.com" is selected as an observed domain. Since there is no user system yet, selected domains will be saved persistently in the users' browser using the "localStorage" object. For the prototype, neither the selection of the primary domain nor of the observed domains is restricted. Any domain the FactServer has data about can be selected. The colors used for the domains and charts can be changed in the settings (cf. section 4.3.3). Figure 4.7 shows the domain picker without any domains selected and with both primary and multiple observed domains selected.

Figure 4.6.: Implementation of the overview of the Analytics Dashboard.

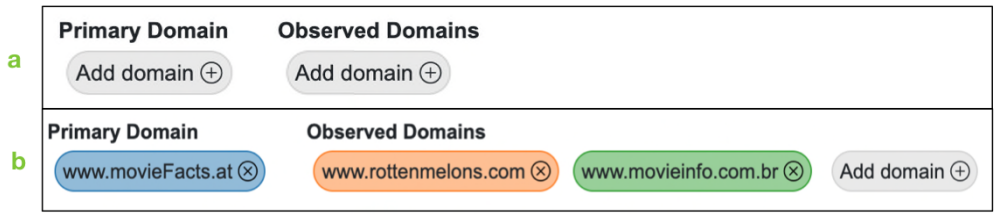

Figure 4.7.: The domain picker without any domains selected (a) and with multiple domains selected (b).

The two date pickers are set to their default value, "last 7 days", for historical data, and "last 90 days", for live data, which means that by default, the instances in the live data were used to calculate the latest historical data. When changing the time unit, the date picker will change the selected value to the closest equivalent of the new unit and offer a matching HTML5 input. Other than the selected domains, it does not make sense to persistently save the date selection since it is likely to change regularly. Therefore, the selected dates are only saved in the SvelteKit session for the session duration (a SvelteKit session is ended by reloading a page). Figure 4.8 shows the date picker in more detail with different time units selected.

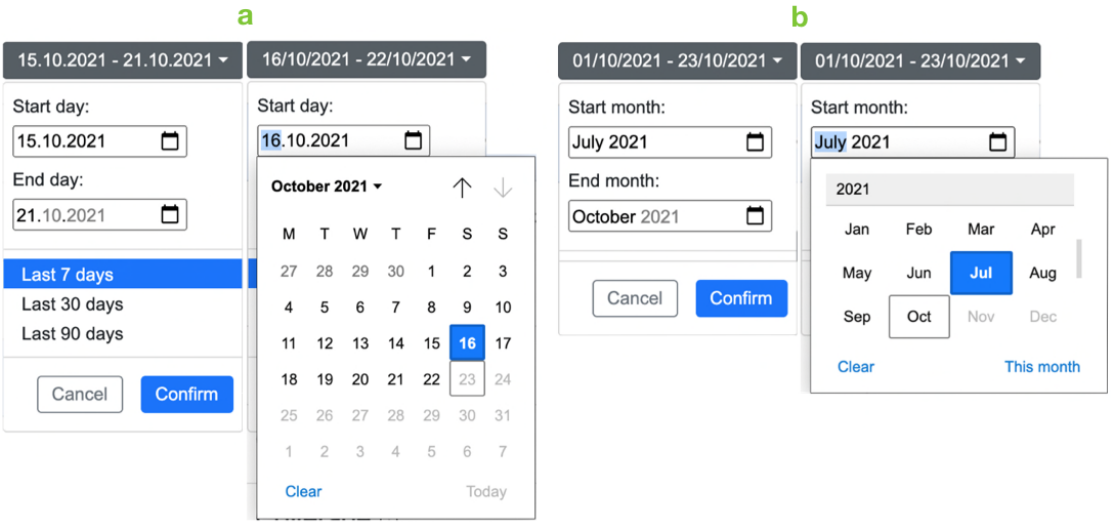

Figure 4.8.: The date picker for the time unit day (a) and time unit month (b).

When the selection of domains changes, all data for the overview is reloaded. A throbber indicates this in the middle of the screen. If the time range changes, only the data that is affected will be reloaded. The interface remains responsive during this time. Live data and historical data are retrieved from the APIs shown in table table 4.2 and table 4.3 respectively.

In figure 4.6 the charts for critical instances and the Page Ranking are visible. The charts of the other metrics are collapsed. Like the selection of domains, the selection of which charts are visible will be saved persistently in the users' browser using the "localStorage" object.

The historical chart for critical instances shows the data in a bar chart with percentage values, and the historical chart for the Page Ranking shows the data in a line chart with the average Page Ranking of the domain. The value type (quantity or percentage) can only be chosen for the critical instances chart since the Page Ranking is already an average of the domain. To prevent labels on the x-axis from overlapping, they are rotated, and if this is not enough, fewer labels are shown. Still, there is a limit of values that can be displayed on the x-axis. It is up to the user to decide if it would make more sense to choose a different time unit to reduce the number of values. Figure 4.9 shows a historical chart with a range of 30 days and how it changes using a different time unit.

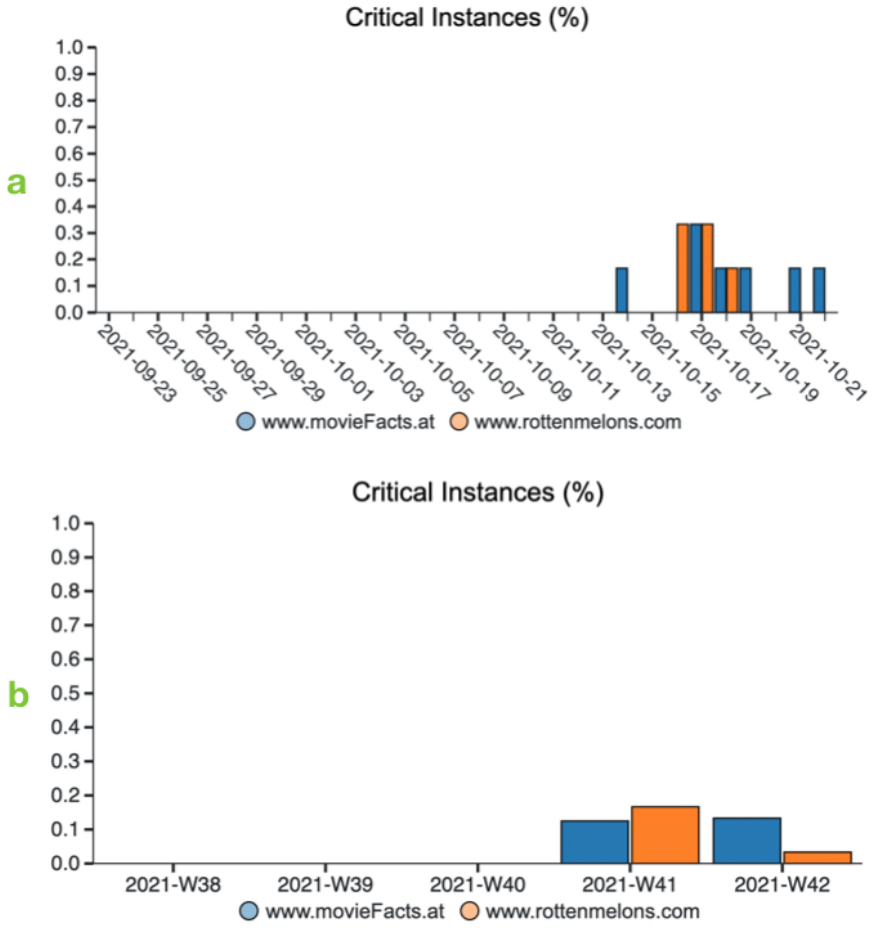

Figure 4.9.: Live data chart showing the critical instances of the last 30 days (a) and an equivalent time span with the time unit "weeks" (b).

Live data charts can handle categorical and quantitative data types. The boolean metrics critical instances, complete instances, and potential rumors, as well as the numeric metric Page Ranking (natural numbers from 1-5) fall in the first category. It is only necessary to count the number of instances in the different categories for these metrics. For the coverage metric, which falls in the latter category, it is still necessary to define buckets. For the prototype, the bucket size is fixed to 0.2 between the 0 to 1 range. Figure 4.6 shows the live data charts for critical instances with percentages and Page Ranking with quantities. Other than historical data charts, live data charts always show the number of instances. Hence the value type can be selected for all of them. Figure 4.10 shows the live data chart for the coverage with its statically set buckets.

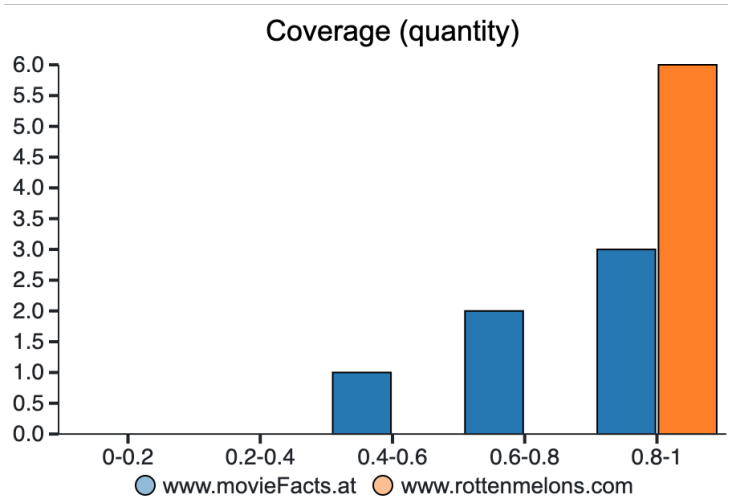

Figure 4.10.: The live data chart for the coverage.

At the bottom of figure 4.6, the live data table can be seen. It was implemented using the Tabulator JavaScript library [21] (version 4.9.3). The figure shows the instances of the last 90 days of the domain "www.movieFacts.at". The filter for critical instances is set only to show non-critical instances, which could have been done by clicking the checkbox in the table header or selecting the "false" bar in the critical instance live data chart. Both approaches will result in the columns showing false critical instances being highlighted, as can be seen in the figure. Like the selection in the date pickers, filters are saved in the SvelteKit session. Pagination was added to avoid that the table becomes too long. Additionally to the name and metrics, a row for the last update date was added so that users can prioritize instances by how recently they were changed. A click on the row will open the single view.

4.3.2. Single View

Figure 4.11 shows the single view of the instance *RoboCop* from the domain "www.movieFacts.at".

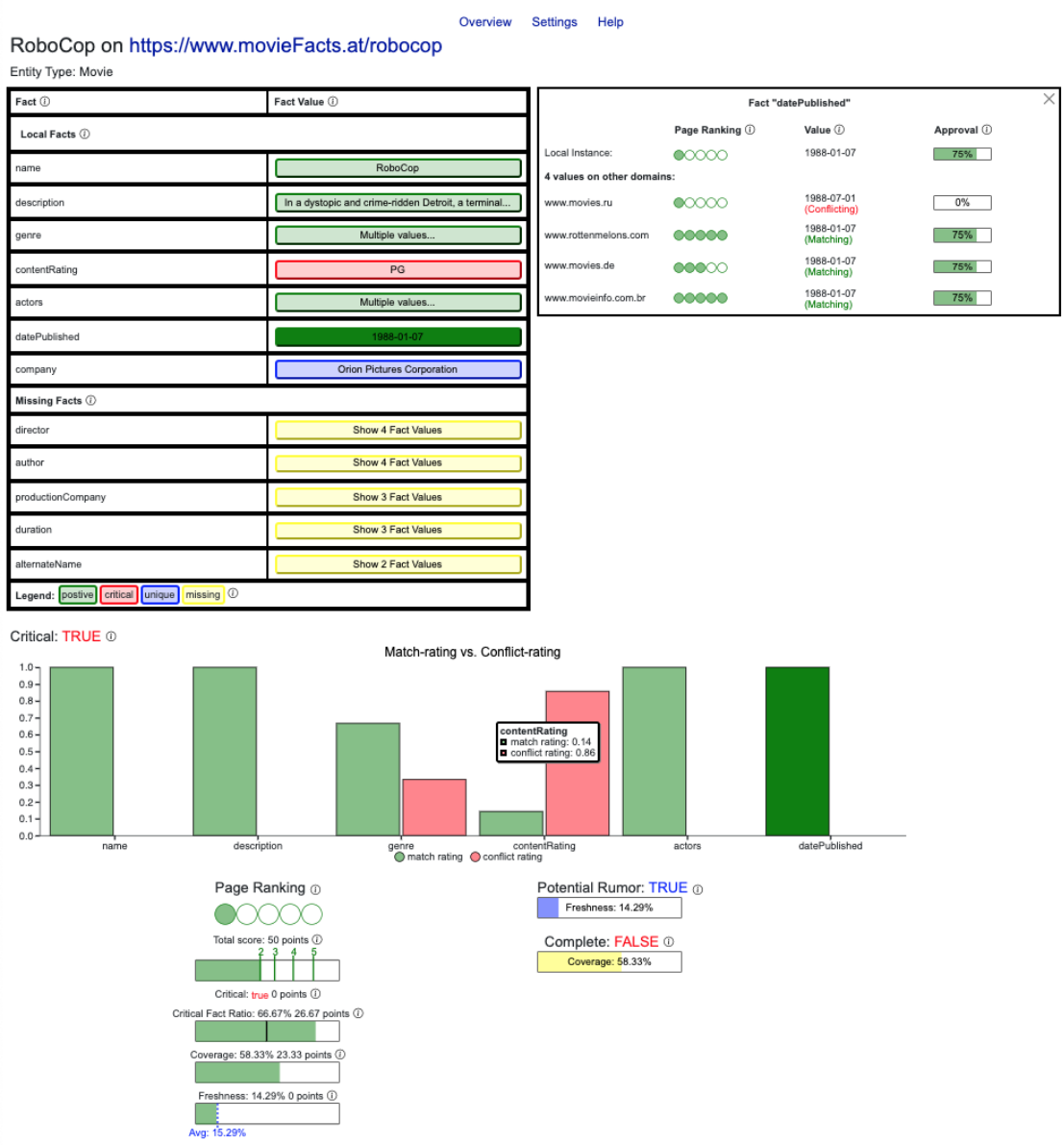

Figure 4.11.: Implementation of the single view of the Analytics Dashboard.

The detailed instance data, retrieved from the API shown in table 4.5, is fetched on the server before the page is sent to the client to avoid loading time.

The fact table shows the list of local and missing facts. If the fact value is not a primitive (i.e., strings, numbers, dates, etc.), for example, a list of values, an image, or a JSON object, it will not be displayed directly in this table to keep the buttons and table short. Instead, as can be seen for the facts "actors" or "genre", the button shows a placeholder. Strings that are too long will also get truncated, as can be seen for the fact "description". The button for missing facts shows the number of domains the fact can be found on rather than a value.

The detailed fact table currently shows the fact "datePublished" which matches the values on all domains except "www.movies.ru". Since the conflicting fact has no matches, it affects the approval but not the match rating, as seen in the critical instance diagram below. Figure 4.12 shows how the detailed fact diagram differs for missing and unique facts.

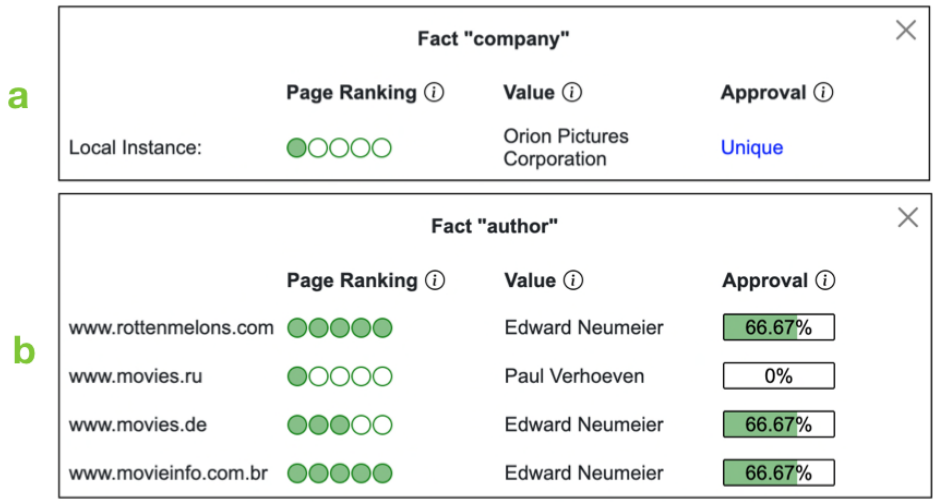

Figure 4.12.: The detailed fact table when a unique fact (a) or missing fact (b) is selected.

As in the fact table, values that are too long are truncated. The full-length value can be seen by clicking the icon at the end of the truncated value. This will open a tooltip showing the full-length value as can be seen in figure 4.13 for the fact "description".

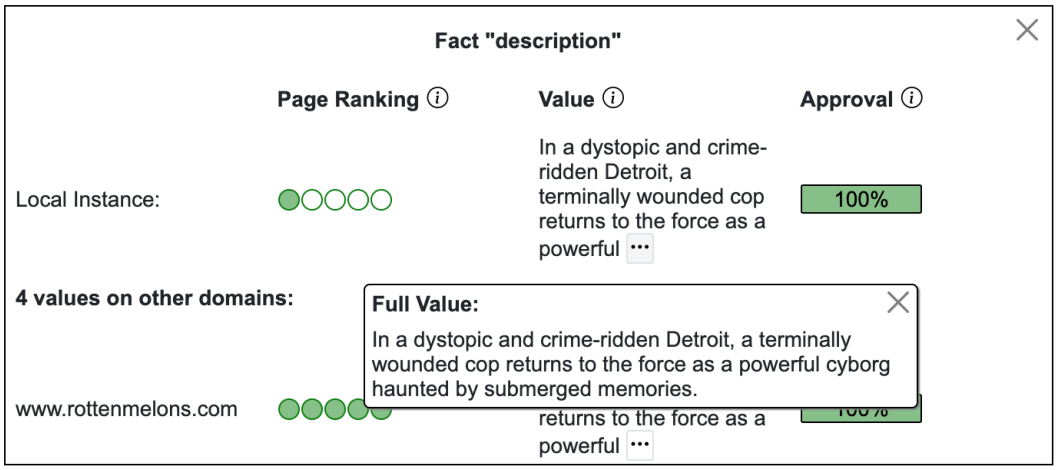

Figure 4.13.: The detailed fact table of the truncated fact "description" with a tooltip showing the full-length fact value.

While lists of primitives are adequately handled, more complex JSON objects are currently only handled like a string with some additional formatting for better readability. This includes, for example, references to other structured data or images. Facts with the name "image" containing only a link to an image are an exception and can be displayed in the table. The solution of showing a formatted representation was chosen since it allows the content provider to see the value as it can be found in the structured data. Modifying the value into a better human-readable form could lead to issues when the value should be changed. Figure 4.14 shows the full-length value tooltip for the fact "director" of an instance where the value is an object with a reference to another structured data.

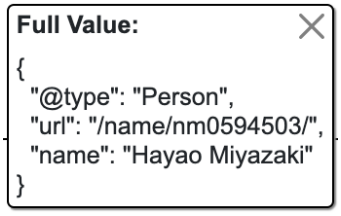

Figure 4.14.: A tooltip showing the full-length fact value consisting of an object. Retrieved from `www.imdb.com/title/tt0096283/`.

The critical instance diagram shows that the current instance is critical. Looking at the chart in figure 4.11, this is due to the fact "contentRating", which has a conflict rating of 0.86 and a match rating of only 0.14. It can also be seen that the bars of the fact "datePublished" are highlighted as it is currently selected.

At the bottom of the view, the Page Ranking diagram can be seen. The current instance has a Page Ranking of 1 with a total of 50 points. They result from 0 points because the instance is critical, 26.67 points for the critical fact ratio, 23.33 points for the coverage,

and 0 points because the instance's freshness is lower than the average. The bars' colors do not change depending on their value, except for the bar for the critical fact ratio, which will turn red if negative points are awarded.

Finally, the potential rumor diagram and complete instance diagram can be seen next to the Page Ranking diagram. The current instance is a potential rumor because of one unique fact and incomplete because five facts are missing.

4.3.3. Settings

A separate settings page was created to allow users to modify the colors used in the dashboard. Colors for the overview can be selected from all categorical color scales included in D3.js, while the colors of the different fact categories can be chosen freely with a color picker. Figure 4.15 shows the default settings and modified settings, including examples of how the overview and single view change.

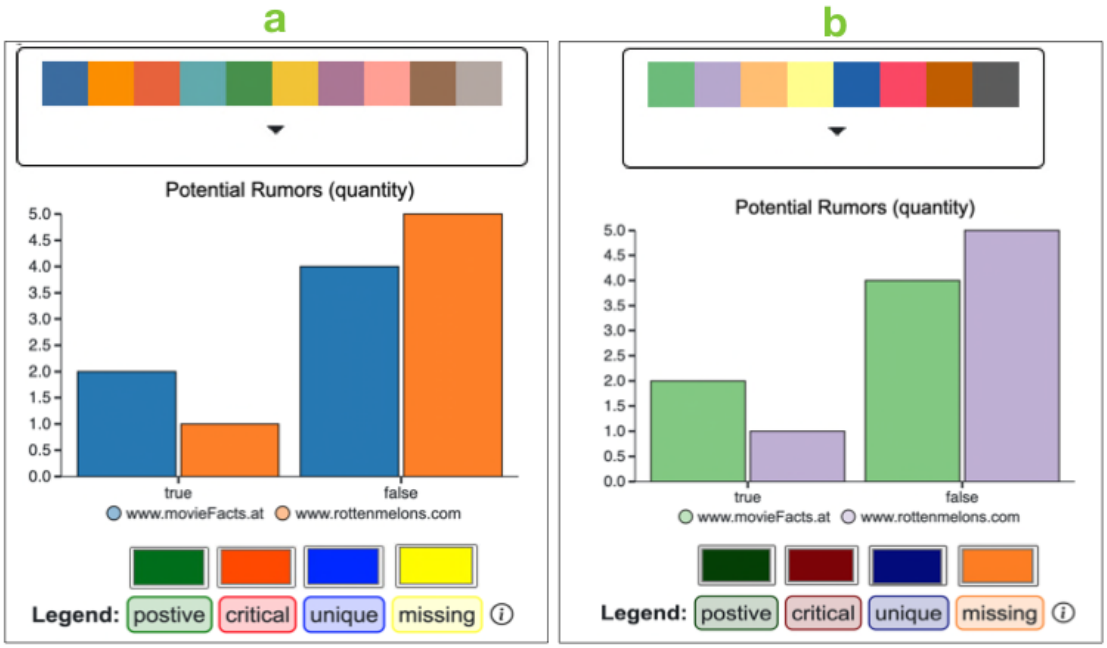

Figure 4.15.: The color settings with examples of how they affect the views. (a) shows the default settings, (b) changed settings.

Since there is no user system yet, the settings will be saved persistently in the users' browser using the "localStorage" object.

4.3.4. Help

As defined in the concept, the Analytics Dashboard provides help to users to understand essential terms with both a dedicated help page and info buttons placed all over the views. The help page can be opened by clicking the "help" link at the top of every view and contains everything worth knowing about the dashboard. The info buttons can be seen both in the overview (cf. figure 4.6) and the single view (cf. figure 4.11). Figure 4.16 shows an example for a tooltip that is shown when clicking on one of the info buttons.

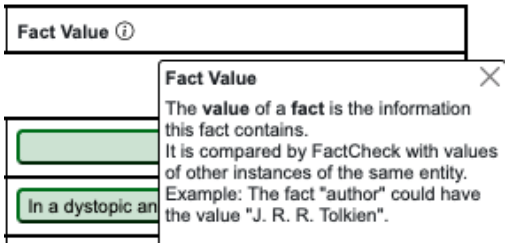

Figure 4.16.: Example for a help tooltip in the implementation of the Analytics Dashboard.

4.3.5. User Analysis

A cookie banner is shown on the first page visit to request permission to use cookies by Google Analytics. If the user accepts, the application will send information about the behavior on the website and the user to Google Analytics. The prototype currently only gathers events automatically collected by Google Analytics and does not send any custom events. Figure 4.17 shows the cookie banner and an example of different statistics that are collected.

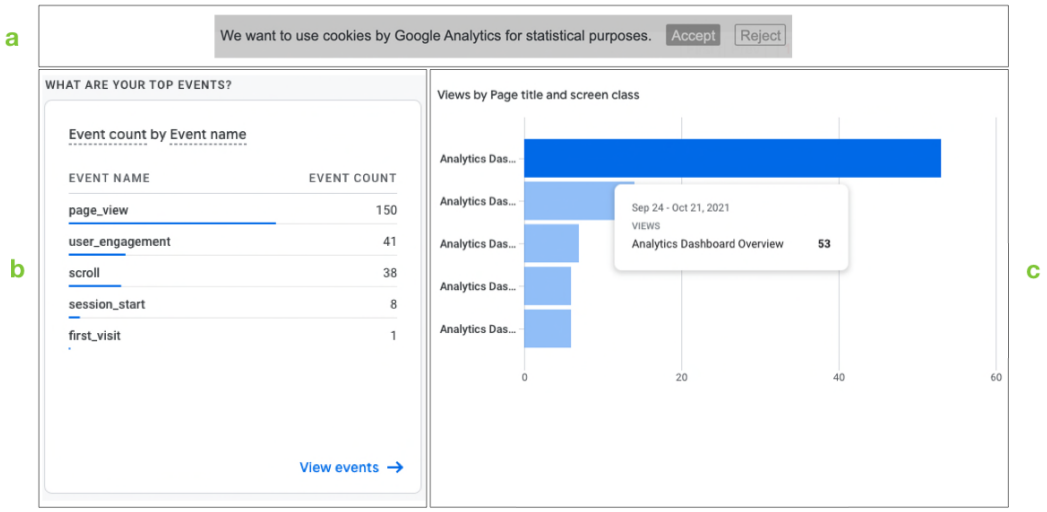

Figure 4.17.: Figure showing the Cookie Banner (a), the top events (b), and the top page views (c) from Google Analytics.

5. Functional Evaluation and Illustrative Use Cases

In this chapter, the prototypical implementation from chapter 4 will be evaluated. Section 5.1 will show that the prototype fulfills the functional requirements defined in section 3.1. The following section 5.2 will then provide examples of how content providers can use the dashboard to find and fix issues in structured data. Finally, section 5.3 will present several limitations of the prototype and the FactCheck++ framework in general. The same data that was used to present the prototype in section 4.3 was used for the evaluation.

5.1. Functional Evaluation

The first functional requirement (**F1**) was that content providers have an overview consisting of visualizations of multiple metrics about misinformation to assess the state of their domain. This requirement was fulfilled by providing three metrics (counting coverage and complete instance once) covering different potential sources of misinformation, a fourth metric combining them into a single rating, and its visual presentation in the overview. Critical instances cover conflicts as a potential source of misinformation, potential rumors bring attention to possibly unverified information, and the coverage and complete instance metrics cover missing facts as a possible origin of misinformation. The fourth metric, Page Ranking, is a combination of those metrics primarily designed for content consumers. However, it also provides a good overview for content providers besides the importance of improving the rating that is directly communicated to their users. Finally, the implementation of the overview (cf. figure 4.6) presents the aggregated metric values of all instances of the domain in the live data charts, allowing content providers to assess their domain according to the different metrics.

The presentation of the metrics in the overview also fulfills the second requirement (**F2**) - content providers can see the daily development of the different domains to identify improvement or deterioration. The regularly collected metric values are presented in the historical data charts allowing content providers to see how they have developed over time. The historical data chart on the left side of figure 5.1 shows this in more detail on the example of the Page Ranking metric. The numeric values of each data point were added for easier interpretation and are not a feature of the prototype. The historical data chart shows that the average Page Ranking for "www.movieFacts.at" starts at 4.7 and decreases over the next five days to a value of 4.17. While this is only a slight decrease over a short time, this trend could still be a warning to content providers.

The third requirement (**F3**) was that content providers can compare their domain with other domains. This requirement was fulfilled by the possibility of adding other domains to the overview. Figure 5.1 shows the observed domain "www.rottenmelons.com" next to the primary domain "www.movieFacts.at". The historical chart on the left allows comparing how values have changed over time on different domains. It shows that the average Page Ranking slightly improves for the observed domain from 4.5 to 5 while the average of the primary domain decreases in the same period. The live data chart on the right allows comparing the current state of different domains. It shows that the observed domain only has instances with a rating of 5, while the primary domain has one instance with a rating of 4 and one instance with a rating of 1. A possible conclusion for the content provider could be that measures against misinformation on the observed domain are more effective. Thus, when comparing fact values, they could prefer values found on the observed domain over others.

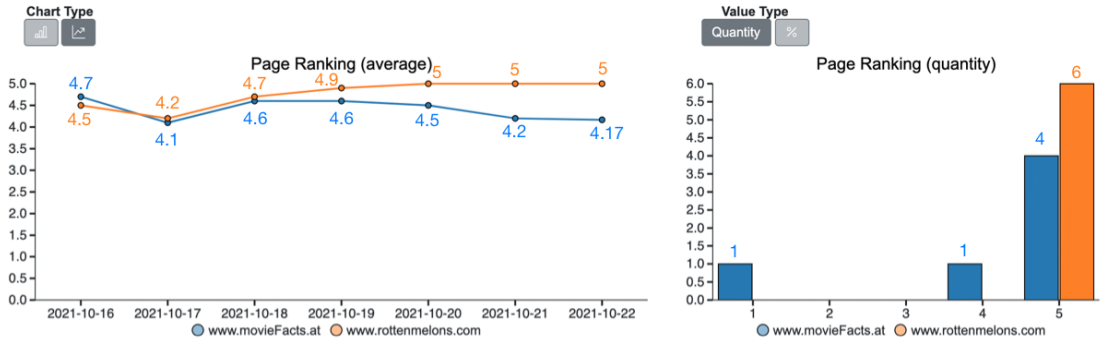

Figure 5.1.: The overview charts for Page Ranking with added values for each data point for easier interpretation.

The fourth requirement (**F4**) was that content providers can also see detailed information about a single instance, which allows them to identify and understand the value for each metric and fix potential issues. This requirement was fulfilled by the presentation of detailed data about single instances in the single view. The view allows content providers to see the fact values of the instance and how they influence the metrics. It also lets them explore and compare values from other domains supporting them in fixing issues. Figure 5.2 shows in detail what parts of the single view are relevant to each metric for understanding its value.

- The value of the *critical instance* metric can be seen in the critical instance diagram (2). The explanation for this metric is the existence or absence of a critical fact in the fact table (1). The critical instance diagram (2) also shows each fact's match and conflict rating to understand why a fact is critical. Finally, the match and conflict rating can further be explored in the detailed fact table, which contains the approval values (3) used to calculate the ratings.

- The value of the *potential rumor* metric can be seen in the potential rumor Diagram (2). The explanation for this metric is the existence or absence of a unique fact in the Fact Table (1). Since a unique fact is defined as exclusive to the local instance, no further explanation is required.

- The value of the *complete instance* and *coverage* metrics can be seen in the complete instance diagram (2). The explanation for this metric is the existence or absence of missing facts in the fact table (1). Since a missing fact is defined as missing in the local instance, no further explanation is required.

- The value of the *Page Ranking* metric can be seen in the Page Ranking diagram (1). The diagram also explains how the Page Ranking is calculated from the other metrics. Since the other metrics are part of this calculation, all previously mentioned parts of the single view can also be seen as part of the explanation for the Page Ranking.

Section 5.2 will present various examples of how content providers can use the dashboard to fix potential issues, further showing how this requirement was fulfilled.

The fifth requirement (**F5**) was that content providers can filter for instances in the overview and then navigate to the single view of instances of interest. While the live data date picker allows filtering instances by their update date, the requirement is primarily fulfilled in the live data table of the overview (cf. figure 4.6). It allows content providers to filter and sort instances by any of the metrics, last updated date, and name. Users can set filters using either the live data charts or the manual filters in the table headers. Clicking on any of the instances in the tables will redirect the user to the single view.

Finally, the sixth requirement (**F6**) was that the service provider can analyze the usage of the Analytics Dashboard. This requirement is fulfilled by using cookies that send data to Google Analytics for further analysis. For the prototype, the data collected automatically by Google Analytics is considered sufficient to fulfill this requirement.

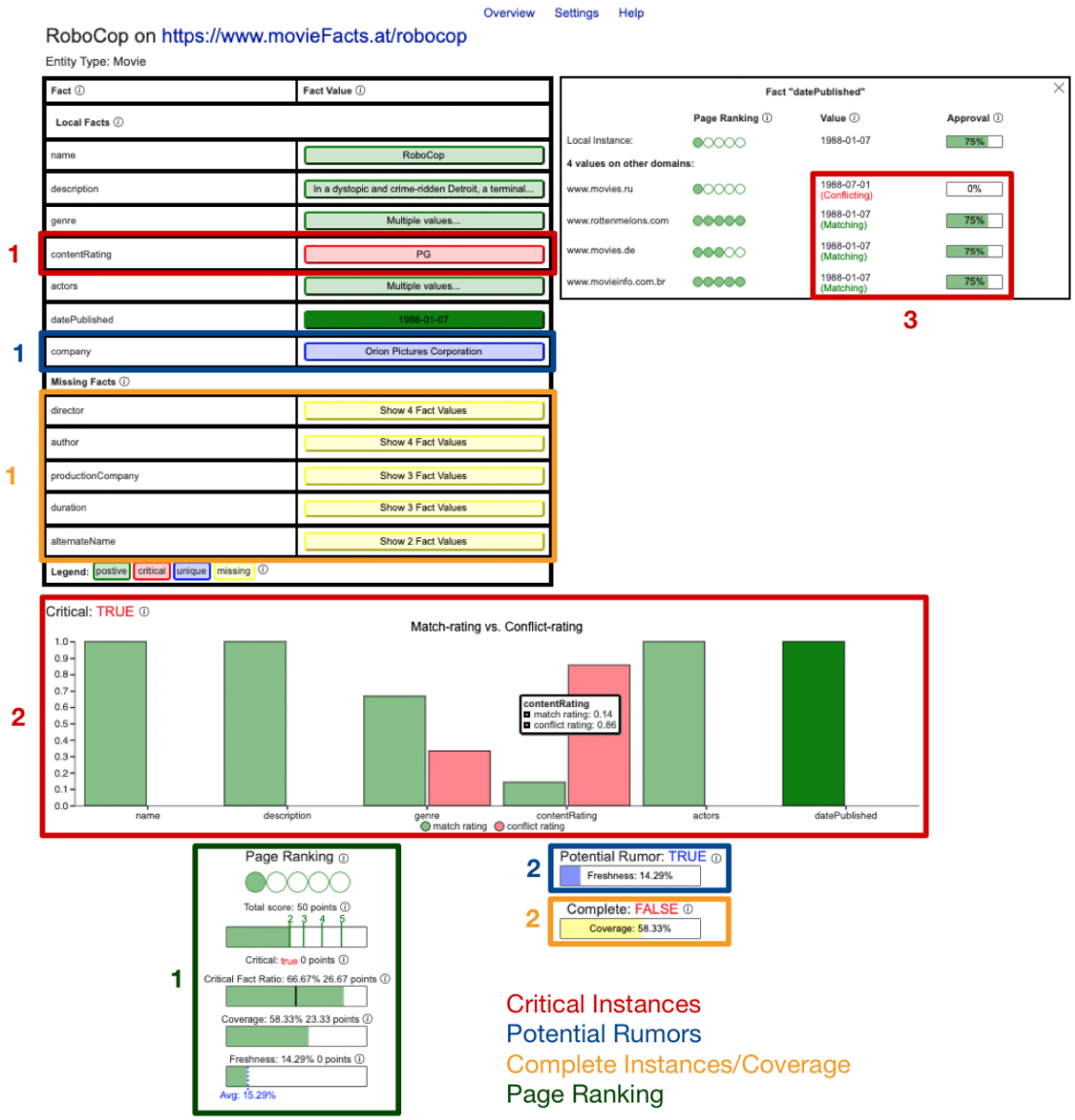

Figure 5.2.: The single view of the Analytics Dashboard with highlights showing their relevance to the metrics.

5.2. Illustrative Use Cases

This section will present three use cases that show how an issue detected in the overview can be fixed with the single view. Each use case will correspond to one of the three metrics conflicting instances, potential rumors, and complete instances and coverage. Fixing those issues will also lead to a better Page Ranking, so an additional use case is not required. It would only be a combination of all three use cases.

5.2.1. Use Case: Fixing a Critical Instance

The following steps are an example of how to fix a critical instance:

1. A new critical instance was detected in the overview. Therefore, the live data table is used to find the instance and open the single view.
2. In the single view, the critical instance diagram or the fact table can be used to find the critical fact "contentRating" (cf. figure 5.2 (1),(2)). Interacting with either diagram opens the detailed fact table.
3. The detailed fact table that opens is shown in figure 5.3. It shows that the local value matches with the domain "www.movies.ru" but conflicts with three other domains all having the same value. Also, the approval bars color shows that the matching value is from a critical instance, while the other instances are non-critical. Thus, it is likely that the current value is false and should be replaced by the conflicting value. Changing the value would make the instance non-critical.

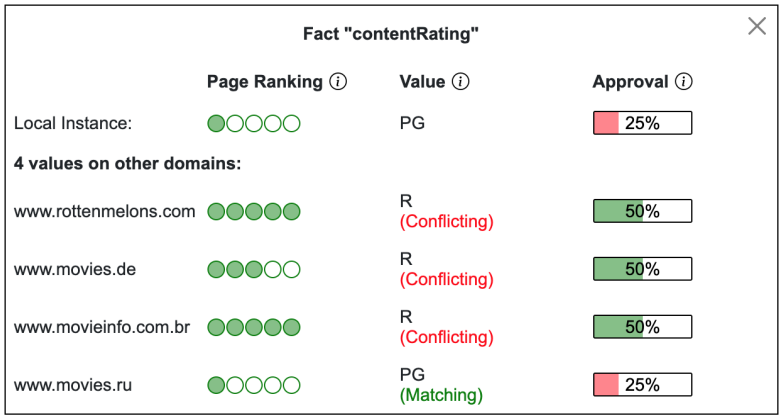

Figure 5.3.: The detailed fact table for the critical fact "contentRating".

5.2.2. Use Case: Completing an Instance

The following steps are an example of how to complete an incomplete instance:

1. A new incomplete instance was detected in the overview. Therefore, the live data table is used to find the instance and open the single view.
2. The fact table shows that five facts are currently missing (cf. figure 5.2 (1)). A value for each of them needs to be added to complete the instance. Clicking on the button in the fact value column opens the detailed fact table, which assists with finding a value.
3. Figure 5.4 shows the detailed fact table for the missing fact "author" as an example how to select a value. The fact exists on four different domains and matches on three of them. Moreover, two of the instances have a Page Ranking of 5. The instance containing the differing value has a Page Ranking of only 1. Therefore, the structured data should be extended by the fact "author" with the value *Edward Neumeier*. Before adding a new fact, their correctness needs to be checked, especially if it only exists on a single domain for its risk of being a rumor. This step is now repeated until no facts are missing.

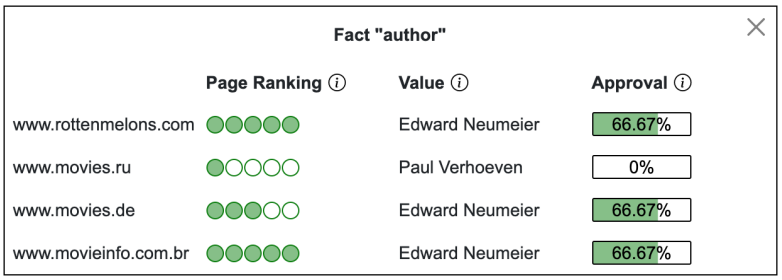

Figure 5.4.: The detailed fact table for the missing fact "author".

5.2.3. Use Case: Checking a Potential Rumor

The following steps are an example of how to check a potential rumor:

1. A new potential rumor was detected in the overview. Therefore, the live data table is used to find the instance and open the single view.
2. The Fact Table shows that the fact "company" of the instance is unique (cf. figure 5.2 (1)). Checking the value for this fact shows that it is correct.

3. Next, the fact name needs to be checked because of the possibility of the value being assigned incorrectly. The fact table can be used to check for facts with a similar name and value. As shown in figure 5.5 in this case a fact "productionCompany" can be found which exists in 3 different instances. It is therefore likely that "company" is the incorrect fact name for the entity. By changing the fact name, the instance is no longer a potential rumor. If the fact name was correct, it is currently not possible to fix a potential rumor.

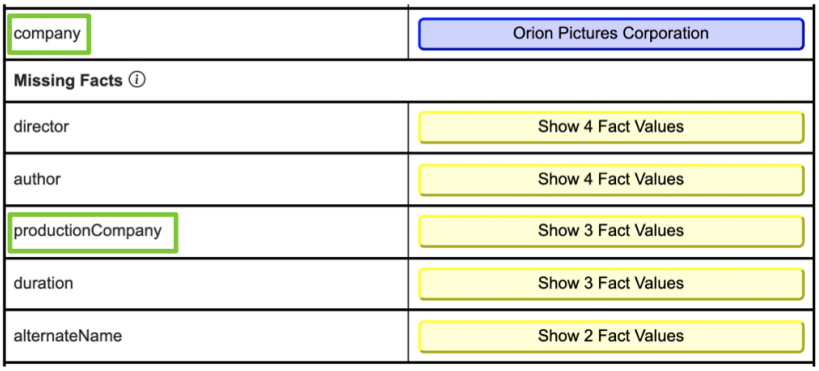

Figure 5.5.: A part of the fact table used to check a potential rumor. The notably similar fact names are highlighted.

5.3. Limitations

This thesis has some limitations due to missing features in the prototype, a short evaluation without a user study, and missing features of the FactCheck++ framework that would allow to improve the prototype. The following sections will present the limitations of this thesis separated into the categories mentioned above.

5.3.1. Limitations of the Prototype

The main limitation of the prototype is that the regular data collection only collects data for instances of the last 90 days, despite the concept for regular data collection suggesting that multiple time spans might be the best solution. This simplification was done to reduce the necessary changes to the FactServer and FactBase where this functionality was implemented. Because of this, users that produce much content might have instances included in the regularly collected data, which are no longer of interest to them, e.g., because it is already archived and no longer available to the public. They would be forced to change old content to improve their ratings. Otherwise, they must accept that the dashboard does not reflect their domain until the last update of the affected content was more than 90 days ago. This issue is amplified since there is no possibility to delete instances that are no longer relevant from the FactBase.

On the other hand, the lack of longer time spans, or even a time span covering all instances existing on the website, makes it more challenging to detect older potential misinformation. While it is possible to choose any time span for the live data independent of the regular data collection, which allows showing instances that are no longer included in the historical data, there is less of an incentive to it.

Another limitation of the prototype is that it can only properly handle primitives, lists of primitives, and images if they follow a specific format. Objects, references to structured data, or other values are only formatted to make them easier to read, which does not tackle the issue that they are not intended for human interpretation. A solution presenting the value in a more user-friendly way while also allowing to examine the raw value was not implemented in the prototype, limiting its usability for those values.

Finally, the collection of user data to analyze the usage of the user dashboard is kept very simple for the prototype. While the default collection of Google Analytics gives some insight, like the number of users, what views they used, and what instances they have checked, other user behavior remains unknown. Information about the usage of diagrams would require custom events, and while this was considered in the concept, it was not implemented in this prototype.

5.3.2. Limitations of the Evaluation

Due to the unavailability of members of the target group, the evaluation of the prototype is limited and did not include a user study as it is usual for similar projects. Therefore, the prototype was only evaluated according to the fulfillment of the requirements and the possibility of accomplishing a selection of tasks for which it was intended. Without a user study with the target group, it is neither possible to evaluate the usability nor if it meets the requirements of a content provider. In this regard, the prototype can only be seen as proof of the possibility to create a dashboard for content providers using the data provided by the FactCheck++ framework.

Another limitation to the evaluation is that no actual data was used. As was argued in section 4.3, a primary reason for this was to control the data used, which would not be possible with actual data. It would be nearly impossible to find an instance that shows all the different concepts while also not being affected by the limitations of the prototype. Another reason no evaluation using actual data was done is that it usually contains complex JSON objects like references to structured data. Handling them is not only a limitation of the prototype (cf. section 5.3.1) but also of the FactCheck++ framework in general, as will be further explained in section 5.3.3. While the predefined data was well suited to present and evaluate the prototype in a controlled environment, it is unknown if the dashboard still fulfills its purpose when used with actual data.

Without using actual data, it was also not possible to evaluate the Page Ranking introduced as a rating primarily intended for content consumers, while also useful to content providers. While other metrics describe the presence of patterns in structured data observed by the FactCheck++ framework, the Page Ranking claims to combine the other metrics into a unified rating that reflects how well an instance is doing in general. This claim should be validated by a user study showing that users generally agree with the rating and by showing that the rating scales correctly with the underlying measures. For example, while it can be said that a higher coverage can be considered to be positive, it is unknown what value can be considered to be good or bad and if a linear scaling does reflect this well due to the lack of actual data. Therefore, the validity of the Page Ranking cannot be confirmed in the thesis.

5.3.3. Limitations of FactCheck++

The main limitation of the FactCheck++ framework at the point of writing is that only comparisons of primitives values, lists of primitives, and simple JSON objects, provide accurate results. Structured data often contains values that the framework cannot handle yet. An example for this can be seen in the fact value of "director" of the structured data retrieved from `www.imdb.com/title/tt0096283/`:

```
{
    "@type":"Person",
    "url":"/name/nm0594503/",
    "name":"Hayao Miyazaki"
}
```

The object references the URL `www.imdb.com/name/nm0594503/` which then contains the detailed structured data for the person *Hayao Miyazaki*. FactCheck++ currently does not consider this information but will only compare the objects' property names and values. This approach will usually result in a conflict unless both instances reference the same object and use the same syntax. While the latter can be expected when content providers use the same standard for structured data, the URL and the referenced structured data cannot be expected to be identical. The result of this is that structured data often has many conflicts which cannot be solved. As the predicates, precision metrics, and data collection of the FactCheck++ framework are further developed, it can be expected that this issue will be solved in the future.

Another limitation is that dashboard users currently have to rely on other users to fix a critical fact in some situations. This can, for example, be necessary when a user wants to resolve a critical fact value that they can prove to be correct. Assuming that the value is critical due to two other fact values that agree on a different, incorrect value, the only possibility for the user is to convince at least one other user to change the value. Relying on other content providers is not a user-friendly solution if communication outside the framework is required. A solution could be to consider user- and expert feedback in classifying critical facts that can either influence or overrule the calculation from matches and conflicts. Currently, no such rating system exists, resulting in this limitation.

This issue is further intensified because the FactCheck++ framework is prone to duplicates, making it more common for a false fact to be identified as non-critical, although it only matches with itself.

The same limitation also affects potential rumors where it is not possible to verify a unique fact and complete instances where it is not possible to mark a fact as irrelevant to the entity, so it does not cause other instances to be considered incomplete.

The next limitation is one reason why the prototypes' regular data collection takes place every day at 00:00:00 UTC, as mentioned in section 4.2.2. Without any information about the domain's timezone provided by the framework, the only possible solution would be to collect data at midnight for every timezone and retrieve data of the users' current timezone. This would not be an optimal solution either since historical data would differ over multiple time zones. The solution of using a single timezone makes the prototype less helpful depending how fare away a user is from UTC.

The last limitation of the prototype caused by the current state of the FactCheck++ framework is that it offers no possibility to request an update of instances. Currently, the only possibility to update them is by visiting the page containing the structured data with the IdaFix Browser plugin or by calling the API endpoint presented in table 4.1 with another tool. While the single view of the Analytics Dashboard provides a link to the source of the instance, allowing users to update the instance in combination with the IdaFix Browser plugin, this is not a user-friendly solution. A possibility to update specific instances or even all instances on a domain was planned in an early concept but had to be removed since the functionality currently does not exist, and adding it would go beyond the scope of this thesis.

6. Conclusion

6.1. Summary

This thesis started with an introduction to the topic of misinformation and the various approaches to fact-checking and challenges in the area of automatic fact-checking in chapter 2. The role content providers play in the fight against misinformation was discussed, and it was argued how a dashboard presenting data about misinformation tailored to them could support them in their effort. This short introduction was followed by an overview of the FactCheck++ framework, of which the prototype developed in this thesis will be a part. The section presented how the already existing parts of the framework detect and resolve conflicts in structured data on the web, and various essential terms were explained. Furthermore, it was argued that the current implementation of the Information Dashboard UI does not meet the requirements of content providers. The chapter also presented various related approaches to analyze and represent data about misinformation. However, none targeted content providers, handled structured data, or allowed them to identify and fix conflicts in their content.

Therefore in chapter 3, requirements for a dashboard that should allow content providers to observe the state of their domain and identify and fix conflicts in structured data were defined. Various metrics potentially valuable for content providers that can be calculated from data about misinformation were presented, focusing on those that can be calculated from the data currently provided by the FactCheck++ framework. Based on the requirements, a concept for a dashboard was presented that would allow content providers to utilize these metrics.

Chapter 4 then explained how the concept was implemented in the different components of the FactCheck++ framework, including what technologies were used. The main focus of the chapter was the newly created Analytics Dashboard which realized the concept for the dashboard and is accessible through `http://fcheck02.cs.univie.ac.at:3000`.

The implementation was then evaluated in chapter 5 by checking if the requirements set in the concept were fulfilled and by showing some use cases of the Analytics Dashboard. While no user study was conducted due to the unavailability of target group members, the evaluation shows that the dashboard can be used to identify and fix conflicts in structured data detected by the FactCheck++ framework. Other limitations of the prototype and the framework were mentioned as well.

The approach to include content providers in the combat against misinformation seems promising, as studies show that the most efficient way of avoiding the spread of misinformation is by correcting it as soon as possible. Supporting them in this effort seems like a logical conclusion. Keeping structured data consistent is one possibility of avoiding misinformation. This thesis presents a prototype to show how this task can be supported by providing metrics and visual support, but future work in this area is required.

6.2. Future Work

In this thesis, the Analytics Dashboard for the FactCheck++ framework was presented, but a user study with target group members is still missing. This study should lead to a better understanding of the requirements of content providers, which can be used for the further development of the prototype and the framework in general. Also, the limitations of the prototype need to be addressed in future iterations of the prototype.

An additional feature that could be added in the future is a notification service for dashboard users. It could make them aware of any noteworthy events like a sudden increase of critical instances.

The limitations of the FactCheck++ framework discussed in this thesis also show many future work opportunities.

Currently, the comparison of the framework can only handle primitives, lists of primitives, and simple JSON objects. Structured data, on the other hand, can include more diverse data like images or references to other structured data. It can also contain data that needs to be handled separately, in the most simple case, not comparing it, like ratings specific to the domain (e.g., user rating of a movie). Future work on the comparison and data collection in the FactCheck++ framework should focus on handling more data types to reduce the number of conflicts. This would simultaneously improve the usability of the dashboard and other UI components.

Future work is also required on the methods to collect, store, and utilize user feedback. The evaluation of the prototype has shown that the resolution of some conflicts cannot rely on the result of automatic comparisons alone. A semi-automatic approach that also collects feedback from regular users, content providers, and possibly verified experts could improve the accuracy of the results considerably.

Those changes would also require a user system to track the source of the feedback. Feedback of regular users is not always reliable, so a user system would allow implementing methods to learn about user reliability. The user system could also allow content providers to set the desired timezone for the regular data collection.

Another focus should be on the collection of structured data. Currently, this is done exclusively by the IdaFix browser plugin. While this can work with a large number of users, it should not be relied on to collect and update instances from domains. A web scraper should be added to the framework that other components can use while also autonomously searching and collecting structured data.

Finally, while the concept focuses on structured data, it can also be applied to any other data about misinformation. Supporting the content provider in the fight against misinformation can be a promising approach since it allows to stop the spread of misinformation early. It is hoped that other fact-checking systems further explore the approach to support content providers in keeping content other than structured data free of misinformation.

Bibliography

[1] Ahmet Aker, Leon Derczynski, and Kalina Bontcheva. 2017. Simple Open Stance Classification for Rumour Analysis. *CoRR* abs/1708.05286 (2017). arXiv:1708.05286 http://arxiv.org/abs/1708.05286

[2] Hunt Allcott, Matthew Gentzkow, and Chuan Yu. 2019. Trends in the diffusion of misinformation on social media. *Research & Politics* 6 (04 2019), 205316801984855. https://doi.org/10.1177/2053168019848554

[3] Shaun Anderson. 2021. How Fast Should A Website Load & How To Speed It Up. Retrieved September 16, 2021 from https://www.hobo-web.co.uk/your-website-design-should-load-in-4-seconds

[4] The Tornado Authors. 2021. Tornado Web Server - Tornado Documentation. https://www.tornadoweb.org/en/stable/

[5] Michele Bedard and Chianna Schoenthaler. 2018. Satire or Fake News: Social Media Consumers' Socio-Demographics Decide. 613–619. https://doi.org/10.1145/3184558.3188732

[6] Christina Boididou, Symeon Papadopoulos, Markos Zampoglou, Lazaros Apostolidis, Olga Papadopoulou, and Ioannis Kompatsiaris. 2017. Detection and visualization of misleading content on Twitter. *International Journal of Multimedia Information Retrieval* 7 (12 2017). https://doi.org/10.1007/s13735-017-0143-x

[7] Bootstrap. 2021. Bootstrap - The most popular HTML, CSS, and JS library in the world. http://tabulator.info/

[8] Mike Bostock. 2021. D3.js - Data-Driven Documents. https://d3js.org/

[9] Bjarte Botnevik, Eirik Sakariassen, and Vinay Setty. 2020. BRENDA: Browser Extension for Fake News Detection. *CoRR* abs/2005.13270 (2020). arXiv:2005.13270 https://arxiv.org/abs/2005.13270

[10] Michael Bronstein, Gordon Pennycook, Adam Bear, and Tyrone Cannon. 2018. Belief in Fake News is Associated with Delusionality, Dogmatism, Religious Fundamentalism, and Reduced Analytic Thinking. *Journal of Applied Research in Memory and Cognition* 8 (10 2018). https://doi.org/10.1016/j.jarmac.2018.09.005

[11] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information Credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*

(Hyderabad, India) *(WWW '11)*. Association for Computing Machinery, New York, NY, USA, 675–684. https://doi.org/10.1145/1963405.1963500

[12] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2013. Predicting information credibility in time-sensitive social media. *Internet Research: Electronic Networking Applications and Policy* 23 (10 2013). https://doi.org/10.1108/IntR-05-2012-0095

[13] Sylvie Cazalens, Philippe Lamarre, Julien Leblay, Ioana Manolescu, and Xavier Tannier. 2018. A Content Management Perspective on Fact-Checking. 565–574. https://doi.org/10.1145/3184558.3188727

[14] Yimin Chen, Nadia Conroy, and Victoria Rubin. 2015. Misleading Online Content: Recognizing Clickbait as "False News". https://doi.org/10.1145/2823465.2823467

[15] Keith Coleman. 2021. Introducing Birdwatch, a community-based approach to misinformation. Retrieved September 16, 2021 from https://blog.twitter.com/en_us/topics/product/2021/introducing-birdwatch-a-community-based-approach-to-misinformation.html

[16] Anouk de Regt, Matteo Montecchi, and Sarah Lord Ferguson. 2019. A false image of health: how fake news and pseudo-facts spread in the health and beauty industry. *Journal of Product & Brand Management* ahead-of-print (08 2019). https://doi.org/10.1108/JPBM-12-2018-2180

[17] Facebook. 2021. *Our Progress Addressing Challenges and Innovating Responsibly*. Retrieved September 21, 2021 from https://about.fb.com/news/2021/09/our-progress-addressing-challenges-and-innovating-responsibly/

[18] FactCheck. 2021. FactCheck. https://www.factcheck.org

[19] Miriam Fernandez and Harith Alani. 2018. Online Misinformation: Challenges and Future Directions. *WWW '18: Companion Proceedings of the The Web Conference 2018*, 595–602. https://doi.org/10.1145/3184558.3188730

[20] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2016. The Rise of Social Bots. *Commun. ACM* 59, 7 (June 2016), 96–104. https://doi.org/10.1145/2818717

[21] Oli Folkerd. 2021. Tabulator. http://tabulator.info/

[22] Apache Software Foundation. 2021. Overview - Apache CouchDB Documentation. https://docs.couchdb.org/en/stable/index.html

[23] Daniel Gerber, Diego Esteves, Jens Lehmann, Lorenz Bühmann, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, and René Speck. 2015. DeFacto-Temporal and Multilingual Deep Fact Validation. *Web Semant.* 35, P2 (Dec. 2015), 85–101. https://doi.org/10.1016/j.websem.2015.08.001

[24] Google. 2021. About Events - Analytics Help. Retrieved October 9, 2021 from https://support.google.com/analytics/answer/9322688?hl=en&utm_id=ad

[25] Google. 2021. Google Analytics. https://analytics.google.com

[26] Google. 2021. Google Knowledge Graph Search API. https://developers.google.com/knowledge-graph

[27] Grammarly. 2021. Grammarly. https://www.grammarly.com

[28] Snopes Media Group. 2021. Snopes. https://www.snopes.com

[29] Alex Grönholm. 2021. Advanced Python Scheduler - ApScheduler 3.8.0. https://apscheduler.readthedocs.io/en/stable/index.html

[30] Zhijiang Guo, Michael Sejr Schlichtkrull, and Andreas Vlachos. 2021. A Survey on Automated Fact-Checking. *CoRR* abs/2108.11896 (2021). arXiv:2108.11896 https://arxiv.org/abs/2108.11896

[31] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster. 1803–1812. https://doi.org/10.1145/3097983.3098131

[32] Elle Hunt. 2016. *What is fake news? How to spot it and what you can do to stop it.* Retrieved September 16, 2021 from https://www.theguardian.com/media/2016/dec/18/what-is-fake-news-pizzagate

[33] Joint statement by WHO, UN, UNICEF, UNDP, UNESCO, UNAIDS, ITU, UN Global Pulse, and IFRC. 2020. *Managing the COVID-19 infodemic: Promoting healthy behaviours and mitigating the harm from misinformation and disinformation.* Retrieved September 16, 2021 from https://www.who.int/news/item/23-09-2020-managing-the-covid-19-infodemic-promoting-healthy-behaviours-and-mitigating-the-harm-from-misinformation-and-disinformation

[34] Peter Kalchgruber, Wolfgang Klas, Nour Jnoub, and Elaheh Momeni. 2018. *FactCheck - Identify and Fix Conflicting Data on the Web.* 312–320. https://doi.org/10.1007/978-3-319-91662-0_25

[35] Wolfang Klas and Peter Kalchgruber. 2017. FactCheck - Identify and Fix Conflicting Data on the Web [Vision]. , 12 pages.

[36] Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. *CoRR* abs/1704.07221 (2017). arXiv:1704.07221 http://arxiv.org/abs/1704.07221

[37] Ryan Mac and Craig Silverman. 2020. *Facebook Live Spread Election Conspiracies And Russian State-Controlled Content Despite Employee Fears.* Retrieved September 16, 2021 from https://www.buzzfeednews.com/article/ryanmac/facebook-live-spread-election-conspiracies-rt

[38] Zlatina Marinova, Denis Teyssou, Nikos Sarris, Jochen Spangenberg, Symeon Papadopoulos, Alexandre Alaphilippe, and Kalina Bontcheva. 2019. WeVerify: Wider and Enhanced Verification for You - Project Overview and Tool Demonstration. https://doi.org/10.36370/tto.2019.23

[39] Jeremy B. Merrill and Will Oremus. 2021. Five points for anger, one for a 'like': How Facebook's formula fostered rage and misinformation. Retrieved November 01, 2021 from https://www.washingtonpost.com/technology/2021/10/26/facebook-angry-emoji-algorithm/

[40] Panagiotis Metaxas, Samantha Finn, and Eni Mustafaraj. 2015. Using TwitterTrails.com to Investigate Rumor Propagation. 69–72. https://doi.org/10.1145/2685553.2702691

[41] Sina Mohseni, Eric D. Ragan, and Xia Hu. 2019. Open Issues in Combating Fake News: Interpretability as an Opportunity. *CoRR* abs/1904.03016 (2019). arXiv:1904.03016 http://arxiv.org/abs/1904.03016

[42] Adam Mosseri. 2016. *Addressing Hoaxes and Fake News.* Retrieved September 16, 2021 from https://about.fb.com/news/2016/12/news-feed-fyi-addressing-hoaxes-and-fake-news/

[43] University of Michigan. 2021. Iffy Quotient - CSMR. https://csmr.umich.edu/projects/iffy-quotient/

[44] Ray Oshikawa, Jing Qian, and William Yang Wang. 2018. A Survey on Natural Language Processing for Fake News Detection. *CoRR* abs/1811.00770 (2018). arXiv:1811.00770 http://arxiv.org/abs/1811.00770

[45] Anjan Pal, Alton Chua, and Dion Goh. 2019. Rumor Analysis & Visualization System.

[46] Gordon Pennycook and David G. Rand. 2019. Fighting misinformation on social media using crowdsourced judgments of news source quality. *Proceedings of the National Academy of Sciences* 116, 7 (2019), 2521–2526. https://doi.org/10.1073/pnas.1806781116 arXiv:https://www.pnas.org/content/116/7/2521.full.pdf

[47] Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait Detection, Vol. 9626. 810–817. https://doi.org/10.1007/978-3-319-30671-1_72

[48] Poynter. 2021. ICFN Code of Principles. Retrieved September 16, 2021 from https://ifcncodeofprinciples.poynter.org

[49] Poynter. 2021. PolitiFact. https://www.politifact.com

[50] Rob Procter, Farida Vis, and Alex Voss. 2013. Reading the riots on Twitter: Methodological innovation for the analysis of big data. *International Journal of Social Research Methodology* 16 (05 2013). https://doi.org/10.1080/13645579.2013.774172

[51] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. 2011. Truthy: Mapping the Spread of Astroturf in Microblog Streams. In *Proceedings of the 20th International Conference Companion on World Wide Web* (Hyderabad, India) *(WWW '11)*. Association for Computing Machinery, New York, NY, USA, 249–252. https://doi.org/10.1145/1963192.1963301

[52] Paul Resnick, Samuel Carton, Souneil Park, Yuncheng Shen, and Nicole Zeffer. 2014. RumorLens: A System for Analyzing the Impact of Rumors and Corrections in Social Media.

[53] Paul Resnick, Aviv Ovadya, and Garlin Gilchrist. 2018. Iffy quotient: A platform health metric for misinformation. *School of Information Center for Social Media Responsability University of Michigan* 1, 10 (2018), 1–10.

[54] Kevin Roose, Mike Isaac, and Sheera Frenkel. 2020. *Facebook Struggles to Balance Civility and Growth*. Retrieved September 16, 2021 from https://www.nytimes.com/2020/11/24/technology/facebook-election-misinformation.html

[55] Jon Roozenbeek, Claudia Schneider, Sarah Dryhurst, John Kerr, Alexandra Freeman, Gabriel Recchia, Anne Marthe van der Bles, and Sander van der Linden. 2020. Susceptibility to misinformation about COVID-19 around the world. *Royal Society Open Science* 7 (10 2020). https://doi.org/10.1098/rsos.201199

[56] Marc Röhlig. 2016. Wahl in Österreich: Google meldete den falschen Gewinner. Retrieved September 24, 2021 from https://www.spiegel.de/netzwelt/web/oesterreich-google-legt-sich-versehentlich-schon-hofer-als-bundespraesident-fest-a-00000000-0003-0001-0000-000000582674

[57] Mike Schmierbach and Anne Oeldorf-Hirsch. 2012. A Little Bird Told Me, So I Didn't Believe It: Twitter, Credibility, and Issue Perceptions. *Communication Quarterly* 60 (07 2012), 317–337. https://doi.org/10.1080/01463373.2012.688723

[58] Chengcheng Shao, Giovanni Ciampaglia, Alessandro Flammini, and Filippo Menczer. 2016. Hoaxy: A Platform for Tracking Online Misinformation. *WWW '16 Companion: Proceedings of the 25th International Conference Companion on World Wide Web* (03 2016). https://doi.org/10.1145/2872518.2890098

[59] Chengcheng Shao, Pik-Mai Hui, Lei Wang, Xinwen Jiang, Alessandro Flammini, Filippo Menczer, and Giovanni Ciampaglia. 2018. Anatomy of an online misinformation

network. *PLOS ONE* 13 (01 2018). https://doi.org/10.1371/journal.pone.0196087

[60] Kai Shu, Deepak Mahudeswaran, and Huan Liu. 2019. FakeNewsTracker: a tool for fake news collection, detection, and visualization. *Computational and Mathematical Organization Theory* 25 (01 Mar 2019). https://doi.org/10.1007/s10588-018-09280-3

[61] Craig Silverman. 2015. Lies, damn lies and viral content. (2015).

[62] Craig Silverman. 2016. *This Analysis Shows How Viral Fake Election News Stories Outperformed Real News On Facebook.* Retrieved September 16, 2021 from https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook

[63] Media Matters Staff. 2016. *Understanding The Fake News Universe.* Retrieved September 16, 2021 from https://www.mediamatters.org/fake-news/understanding-fake-news-universe

[64] SvelteKit. 2021. SvelteKit - The fastest way to build Svelte apps. https://kit.svelte.dev/

[65] Full Fact Team. 2017. £350 million EU claim "a clear misuse of official statistics. Retrieved September 29, 2021 from https://fullfact.org/europe/350-million-week-boris-johnson-statistics-authority-misuse/

[66] NewsGuard Technologies. 2021. NewsGuard Frontpage. https://www.newsguardtech.com/

[67] NewsGuard Technologies. 2021. NewsGuard: Rating Process and Criteria. Retrieved September 16, 2021 from https://www.newsguardtech.com/ratings/rating-process-criteria/

[68] NewsGuard Technologies. 2021. NewsGuard: Why should you trust us? Retrieved September 16, 2021 from https://www.newsguardtech.com/about/why-should-you-trust-us/

[69] Sebastian Tschiatschek, Adish Singla, Manuel Gomez Rodriguez, Arpit Merchant, and Andreas Krause. 2018. Fake News Detection in Social Networks via Crowd Signals. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 517–524. https://doi.org/10.1145/3184558.3188722

[70] Claire Wardle. 2017. *Fake news. It's complicated.* Retrieved September 16, 2021 from https://firstdraftnews.org/latest/fake-news-complicated/

[71] Claire Wardle and Hossein Derakhshan. 2017. *INFORMATION DISORDER : Toward an interdisciplinary framework for research and policy making Information Disorder Toward an interdisciplinary framework for research and policymaking.*

[72] Wordpress. 2021. Wordpress Dashboard Screen. https://wordpress.com/support/stats/

[73] Fan Yang, Shiva Pentyala, Sina Mohseni, Mengnan Du, Hao Yuan, Rhema Linder, Eric Ragan, Shuiwang Ji, and Xia Hu. 2019. XFake: Explainable Fake News Detector with Visualizations. 3600–3604. https://doi.org/10.1145/3308558.3314119

[74] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2017. A Convolutional Approach for Misinformation Identification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 3901–3907.

[75] Amy X. Zhang, Aditya Ranganathan, Sarah Emlen Metz, Scott Appling, Connie Moon Sehat, Norman Gilmore, Nick B. Adams, Emmanuel Vincent, Jennifer Lee, Martin Robbins, Ed Bice, Sandro Hawke, David Karger, and An Xiao Mina. 2018. A Structured Response to Misinformation: Defining and Annotating Credibility Indicators in News Articles *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 603–612. https://doi.org/10.1145/3184558.3188731

[76] Lina Zhou and Dongsong Zhang. 2007. An Ontology-Supported Misinformation Model: Toward a Digital Misinformation Library. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 37, 5 (2007), 804–813. https://doi.org/10.1109/TSMCA.2007.902648

[77] Xinyi Zhou, Atishay Jain, Vir V. Phoha, and Reza Zafarani. 2020. Fake News Early Detection: A Theory-Driven Model. *Digital Threats: Research and Practice* 1, 2, Article 12 (June 2020), 25 pages. https://doi.org/10.1145/3377478

[78] Xinyi Zhou and Reza Zafarani. 2020. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. 53, 5, Article 109 (Sept. 2020), 40 pages. https://doi.org/10.1145/3395046

[79] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and Resolution of Rumours in Social Media: A Survey. *ACM Comput. Surv.* 51, 2, Article 32 (Feb. 2018), 36 pages. https://doi.org/10.1145/3161603

A. Appendix

A.1. Source Code

The source code of the prototype can be found on the GitLab server of the University of Vienna. Each of the components is in a separate repository:

- *FactBase:* `https://git01lab.cs.univie.ac.at/factcheck/factbase`
  branch: analytics_dashboard_branch
- *Backend:* `https://git01lab.cs.univie.ac.at/factcheck/backend`
  branch: master
- *FactServer:* `https://git01lab.cs.univie.ac.at/factcheck/factserver`
  branch: analytics_dashboard_branch
- *Analytics Dashboard:* `https://git01lab.cs.univie.ac.at/thesis/2021ss/analyticsdashboard`
  branch: master

A.2. Installation Instruction

The Analytics Dashboard depends on the FactServer, which itself relies on the Backend and FactBase. Each of the following section contains the instructions to the corresponding repository mentioned previously. The instructions can also be found in the README of the repositories. It is recommended to follow them in the order they are mentioned.

A.2.1. FactBase

Python Dependencies

- requests (∼=2.22.0)
- CouchDB (∼=1.2)

Install CouchDB

- `apt-get install couchdb`
- Open /etc/couchdb/default.ini and change bind_address to 0.0.0.0
  `bind_address = 0.0.0.0`

- Allow CORS for couchdb:
  - add three lines to /etc/couchdb/etc/local.ini of couchdb

```
[httpd]
enable_cors = true

[cors]
origins = *
```

- Restart CouchDB `service couchdb restart`
- Optional Set CouchDB Admin user:
  - `curl -s -X PUT http://localhost:5984/_config/admins/<user> -d <password>`
  - `service couchdb` restart

Import Views

- Copy config/default.yaml to config/myconfig.yaml and modify config according to your CouchDB settings.
- Execute the script import_views.py `python3 import_views.py`

A.2.2. Backend

Install Server

- Download Source or GIT Repository
- Rename local_settings-default.py to local_settings.py
  - Modify database access parameters to backend database: MySQL (recommended), ISQL or Postgres(Heroku) in local_settings.py
  - Modify database access parameters to FactBase

Install django

```
apt-get install python-pip -y
pip install "django<2"
```

Install MySQL-Server

```
sudo apt-get install mysql-server python-dev libmysqlclient-dev -y
pip install MySQL-python #whitenoise dj-database-url
```

Create Backend Database

```
echo "CREATE DATABASE backend" | mysql -u root -p;
python manage.py makemigrations
python manage.py migrate
```

Run the BackendServer

```
python backend/manage.py runserver
```

A.2.3. FactServer

Prerequirements

- CouchDB needs to be installed/accessible. The CouchDB Views need to be imported (see A.2.1).
- Backend must be installed/accessible in order to run a FactServer (see A.2.2).

Configure Server

- Copy config/default.yaml to config/myconfig.yaml and modify config according to your local settings and preferences.
- Configure SSL settings in myconfig.yaml (if needed).

Python Dependencies

- APScheduler ∼=3.7.0
- CouchDB ∼=1.2
- python-dateutil ∼=2.8.1
- PyYAML ∼=5.3.1
- tornado ∼=6.1

Run the server

```
python3 server.py
```

Additional Information

- Bulk imported factsets must be added to factstats database by manually running the script factserver/update.py
- In addition, this script allows to recurrently recalculate all factstats by executing the script's update function.

A.2.4. Analytics Dashboard

Prerequirements

- The FactServer needs to be installed and accessible (see A.2.3).
- node.js needs to be installed (compatible with 6.14.13).

Configure the Analytics Dashboard

- modify the adapter in AnalyticsDashboard/svelte.config.js if required (e.g. port or host).
- change the FACT_SERVER environment variable in AnalyticsDashboard/.env to your FactServer.

Install the Analytics Dashboard

- Move to the folder AnalyticsDashboard.
- Install all dependencies `npm install`.
- Build the node.js application `npm run build`.

Start the Analytics Dashboard `node build` (assuming default configuration).

A.3. API Response Examples

A.3.1. Comparison Result

A shortened example for the request `<FactServer>/get`.

```
[
    #comparison between www.movieFacts.at and www.movieinfo.com.br
    {
        "base_url":"https://www.movieFacts.at/robocop",
        "exturl":"https://www.movieinfo.com.br/robocop",
        "equals":{
            "name":[["RoboCop","RoboCop"]],
            "actors":[
                [["Peter Weller","Nancy Allen", "Ronny Cox"],
                ["Peter Weller","Nancy Allen","Ronny Cox"]]
            ],
            "genre":[[["Sci-fi","Action"],["Sci-fi","Action"]]],
            "datePublished":[["1988-01-07","1988-01-07"]],
            "description":[["<description>","<description>"]]
        },
        "local_missing":{
            "alternateName":["RoboCop - Das Gesetz in der Zukunft"],
            "director":["Paul Verhoeven"],
```

```
            "author":["Edward Neumeier"],
            "productionCompany":["Orion Pictures Corporation"],
            "duration":["1H42M"] },
        "remote_missing":{
            "company":["Orion Picture Corporation"]
        },
        "conflicting":{
            "contentRating":[["PG","R"]]
        },
        "pm":{
            "name":"Default",
            "actors":"Default",
            "contentRating":"Default",
            "genre":"Default",
            "datePublished":"Default",
            "description":"Default"
        },
        "mid":"kg:/m/0jzfgh7"
    },
    #comparison with other instances
    ...
]
```

The following body was sent as a parameter:

```
{
    "data": {
        "@context": "http://schema.org",
        "@type": "Movie",
        "name": "RoboCop",
        "actors": ["Peter Weller", "Nancy Allen", "Ronny Cox"],
        "contentRating": "PG",
        "genre": ["Sci-fi", "Action"],
        "company": "Orion Pictures Corporation",
        "datePublished": "1988-01-07",
        "description": "In a dystopic and crime-ridden Detroit, a
    terminally wounded cop returns to the force as a powerful cyborg
    haunted by submerged memories."
    },
    "source": "https://www.movieFacts.at/robocop",
    "userid":<userID>
}
```

A.3.2. Live Data

A shortened example for the request `<FactServer>/stats?domains=["www.movieFacts.at","www.rottenmelons.com"]&end_date=2021-10-30&start_date=2021-10-15`.

```
{
  "www.movieFacts.at": [
```

```
    {
      "id": "cb7fce9d0135beaec51662f2c703288c",
      "name": "My Neighbor Totoro",
      "source": "https://www.movieFacts.at/totoro",
      "last_updated": "2021-10-14T20:29:26.253455",
      "stats": {
        "num_compared_properties": 10,
        "num_compared_entries": 1,
        "eqconf_ratio": 0.9,
        "num_eq_prop": 9,
        "num_co_prop": 1,
        "num_cmi_prop": 0,
        "num_rmi_prop": 0,
        "num_all_local_prop": 10,
        "num_all_ww_prop": 10,
        "num_unique_facts": 0,
        "potential_rumor": false,
        "freshness": 0,
        "loc_coverage": 1,
        "complete_instance": true,
        "critical_instance": false,
        "page_ranking": 5
      }
    },
    #more instances from www.movieFacts.at
    ...
  ],
  "www.rottenmelons.com": [
    {
      "id": "cb7fce9d0135beaec51662f2c701995b",
      "name": "RoboCop",
      "source": "https://www.rottenmelons.com/robocop",
      "last_updated": "2021-10-14T19:19:36.231140",
      "stats": {
        "num_compared_properties": 11,
        "num_compared_entries": 4,
        "eqconf_ratio": 0.7651515151515152,
        "num_eq_prop": 10,
        "num_co_prop": 5,
        "num_cmi_prop": 1,
        "num_rmi_prop": 5,
        "num_all_local_prop": 11,
        "num_all_ww_prop": 12,
        "num_unique_facts": 0,
        "potential_rumor": false,
        "freshness": 0.20454545454545456,
        "loc_coverage": 0.9166666666666666,
        "complete_instance": false,
        "critical_instance": false,
```

```
        "page_ranking": 5
      }
    },
    #more instances from "www.rottenmelons.com"
    ...
  ]
}
```

A.3.3. Historical Data

A shortened example for the request `<FactServer>/historical?start_date=2021-10-15&domains=["www.movieFacts.at","www.rottenmelons.com"]`.

```
{
  "2021-10-15": {
    "www.movieFacts.at": {
      "eqconf_ratio": 0.8978114478114478,
      "num_compared_properties": 9.833333333333332,
      "freshness": 0,
      "num_eq_prop": 8.833333333333332,
      "num_all_local_prop": 9.833333333333332,
      "num_co_prop": 1,
      "num_cmi_prop": 0,
      "num_rmi_prop": 0,
      "num_all_ww_prop": 9.833333333333332,
      "num_unique_facts": 0,
      "loc_coverage": 1,
      "num_potential_rumor": 0,
      "num_complete_instances": 6,
      "num_critical_instances": 0,
      "page_ranking": 5,
      "sum": 6,
      "num_new_critical_instances": 0
    },
    "www.rottenmelons.com": {
      "eqconf_ratio": 0.896969696969697,
      "num_compared_properties": 9.75,
      "freshness": 0,
      "num_eq_prop": 8.75,
      "num_all_local_prop": 9.75,
      "num_co_prop": 1,
      "num_cmi_prop": 0,
      "num_rmi_prop": 0,
      "num_all_ww_prop": 9.75,
      "num_unique_facts": 0,
      "loc_coverage": 1,
      "num_potential_rumor": 0,
      "num_complete_instances": 6,
      "num_critical_instances": 0,
```

```
      "page_ranking": 5,
      "sum": 6,
      "num_new_critical_instances": 0
    }
  },
  #historical data of other dates
  ...
```

A.3.4. Detailed Data

A shortened example for the request to `<FactServer>/stats/cb7fce9d0135beaec51662f2c701995b`. It returns the detailed data for the id *cb7fce9d0135beaec51662f2c701995b*, which is *RoboCop* on the domain "www.rottenmelons.com".

Request:

```
{
  "num_compared_properties": 11,
  "num_compared_entries": 4,
  "eqconf_ratio": 0.7651515151515152,
  "num_eq_prop": 10,
  "num_co_prop": 5,
  "num_cmi_prop": 1,
  "num_rmi_prop": 5,
  "num_all_local_prop": 11,
  "num_all_ww_prop": 12,
  "num_unique_facts": 0,
  "potential_rumor": false,
  "freshness": 0.20454545454545456,
  "loc_coverage": 0.9166666666666666,
  "complete_instance": false,
  "fact_stats": {
    "contentRating": {
      "approval": 0.5,
      "unique_fact": false,
      "match_rating": 0.6666666666666666,
      "conflict_rating": 0.3333333333333333,
      "critical_fact": false,
      "value": "R",
      "eq_facts": {
        "https://www.movies.de/robocop": {
          "approval": 0.5,
          "unique_fact": false,
          "match_rating": 0.6666666666666666,
          "conflict_rating": 0.3333333333333333,
          "critical_fact": false,
          "value": "R"
        },
        "https://www.movieinfo.com.br/robocop": {
          "approval": 0.5,
```

```
          "unique_fact": false,
          "match_rating": 0.6666666666666666,
          "conflict_rating": 0.3333333333333333,
          "critical_fact": false,
          "value": "R"
        }
      },
      "co_facts": {
        "https://www.movieFacts.at/robocop": {
          "approval": 0.25,
          "unique_fact": false,
          "match_rating": 0.14285714285714285,
          "conflict_rating": 0.8571428571428571,
          "critical_fact": true,
          "value": "PG"
        },
        "https://www.movies.ru/robocop": {
          "approval": 0.25,
          "unique_fact": false,
          "match_rating": 0.14285714285714285,
          "conflict_rating": 0.8571428571428571,
          "critical_fact": true,
          "value": "PG"
        }
      }
    },
    #fact_stats of other facts
    ...
},
"critical_instance": false,
"page_ranking": 5,
"source": "https://www.rottenmelons.com/robocop",
"type": "Movie",
"missing_fact_stats": {
  "company": {
    "https://www.movieFacts.at/robocop": {
      "unique_fact": true,
      "value": "Orion Pictures Corporation"
    }
  }
},
"potential_rumor_other": {
  "https://www.movieFacts.at/robocop": true,
  "https://www.movies.ru/robocop": false,
  "https://www.movies.de/robocop": false,
  "https://www.movieinfo.com.br/robocop": false
},
"freshness_other": {
  "https://www.movieFacts.at/robocop": 0.14285714285714285,
```

```
    "https://www.movies.ru/robocop": 0.0625,
    "https://www.movies.de/robocop": 0.15000000000000002,
    "https://www.movieinfo.com.br/robocop": 0.20454545454545456
  },
  "loc_coverage_other": {
    "https://www.movieFacts.at/robocop": 0.5833333333333334,
    "https://www.movies.ru/robocop": 0.6666666666666666,
    "https://www.movies.de/robocop": 0.8333333333333334,
    "https://www.movieinfo.com.br/robocop": 0.9166666666666666
  },
  "complete_instance_other": {
    "https://www.movieFacts.at/robocop": false,
    "https://www.movies.ru/robocop": false,
    "https://www.movies.de/robocop": false,
    "https://www.movieinfo.com.br/robocop": false
  },
  "critical_instance_other": {
    "https://www.movieFacts.at/robocop": true,
    "https://www.movies.ru/robocop": true,
    "https://www.movies.de/robocop": true,
    "https://www.movieinfo.com.br/robocop": false
  },
  "page_ranking_other": {
    "https://www.movieFacts.at/robocop": 1,
    "https://www.movies.ru/robocop": 1,
    "https://www.movies.de/robocop": 3,
    "https://www.movieinfo.com.br/robocop": 5
  },
  #remaining statistics of other domains
  ...
}
```

A.4. Predefined Structured Data

RoboCop on "www.rottenmelons.com" and "www.movieinfo.com.br"

```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "RoboCop",
  "alternateName": "RoboCop - Das Gesetz in der Zukunft",
  "actors": ["Peter Weller", "Nancy Allen", "Ronny Cox"],
  "director": "Paul Verhoeven",
  "author": "Edward Neumeier",
  "contentRating": "R",
  "genre": ["Sci-fi", "Action"],
  "productionCompany": "Orion Pictures Corporation",
  "datePublished": "1988-01-07",
```

```
  "duration": "1H42M",
  "description": "In a dystopic and crime-ridden Detroit, a
   terminally wounded cop returns to the force as a powerful cyborg
   haunted by submerged memories."
}
```


Robocop on "www.movieFacts.at"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "RoboCop",
  "actors": ["Peter Weller", "Nancy Allen", "Ronny Cox"],
  "contentRating": "PG",
  "genre": ["Sci-fi", "Action"],
  "company": "Orion Pictures Corporation",
  "datePublished": "1988-01-07",
  "description": "In a dystopic and crime-ridden Detroit, a
   terminally wounded cop returns to the force as a powerful cyborg
   haunted by submerged memories."
}
```


Robocop on "www.movies.ru"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "RoboCop",
  "actors": ["Peter Weller", "Nancy Allen", "Ronny Cox"],
  "director": "Paul Verhoeven",
  "author": "Paul Verhoeven",
  "contentRating": "PG",
  "genre": ["Sci-fi", "Action", "Fantasy"],
  "datePublished": "1988-07-01",
  "description": "In a dystopic and crime-ridden Detroit, a
   terminally wounded cop returns to the force as a powerful cyborg
   haunted by submerged memories."
}
```


Robocop on "www.movies.de"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "RoboCop",
  "actors": ["Peter Weller", "Nancy Allen", "Ronny Cox"],
  "director": "Paul Verhoeven",
  "author": "Paul Verhoeven",
  "contentRating": "PG",
  "genre": ["Sci-fi", "Action", "Fantasy"],
  "datePublished": "1988-07-01",
```

```
  "description": "In a dystopic and crime-ridden Detroit, a
   terminally wounded cop returns to the force as a powerful cyborg
   haunted by submerged memories."
}
```


Lord of the Rings on "www.rottenmelons.com" and "www.movieinfo.com.br"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "The Lord of the Rings: The Return of the King",
  "alternateName": "Der Herr der Ringe: Die Rueckkehr des Koenigs",
  "actors": ["Elijah Wood", "Ian McKellen", "Liv Tyler", "Viggo
   Mortensen"],
  "director": "Peter Jackson",
  "contentRating": "PG-13",
  "genre": ["Adventure", "Fantasy"],
  "productionCompany": "New Line Cinema",
  "datePublished": "2003-12-17",
  "duration": "3H21M",
  "description": "Gandalf and Aragorn lead the World of Men against
   Sauron's army to draw his gaze from Frodo and Sam as they approach
    Mount Doom with the One Ring."
}
```


Lord of the Rings on "www.movieFacts.at"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "The Lord of the Rings: The Return of the King",
  "alternateName": "Der Herr der Ringe: Die Rueckkehr des Koenigs",
  "actors": ["Elijah Wood", "Ian McKellen", "Liv Tyler", "Viggo
   Mortensen"],
  "director": "Peter Jackson",
  "contentRating": "PG-13",
  "genre": ["Adventure", "Fantasy"],
  "datePublished": "2003-12-17",
  "duration": "3H21M"
}
```


My Neighbor Totoro on "www.movieFacts.at" and "www.rottenmelons.com"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "My Neighbor Totoro",
  "alternateName": "Mein Nachbar Totoro",
  "actors": ["Hitoshi Takagi", "Noriko Hidaka", "Chika Sakamoto"],
  "director": "Hayao Miyazaki",
  "contentRating": "G",
  "genre": ["Anime", "Family", "Fantasy"],
```

```
  "productionCompany": "Studio Ghibli",
  "datePublished": "1988-04-16",
  "duration": "1H26M",
  "description": "This acclaimed animated tale by director Hayao
   Miyazaki follows schoolgirl Satsuke and her younger sister, Mei,
   as they settle into an old country house with their father and
   wait for their mother to recover from an illness in an area
   hospital. As the sisters explore their new home, they encounter
   and befriend playful spirits in their house and the nearby forest,
    most notably the massive cuddly creature known as Totoro."
}
```


Minions on "www.movieFacts.at" and "www.rottenmelons.com"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "Minions",
  "actors": ["Sandra Bullock", "Jon Hamm", "Michael Keaton"],
  "director": ["Kyle Balda","Pierre Coffin"],
  "contentRating": "PG",
  "genre": ["Comedy", "Animation", "Adventure"],
  "productionCompany": "Illumination Entertainment",
  "datePublished": "2015-07-02",
  "duration": "1H31M",
  "description": "Evolving from single-celled yellow organisms at the
   dawn of time, Minions live to serve, but find themselves working
   for a continual series of unsuccessful masters, from T. Rex to
   Napoleon. Without a master to grovel for, the Minions fall into a
   deep depression. But one minion, Kevin, has a plan; accompanied by
    his pals Stuart and Bob, Kevin sets forth to find a new evil boss
    for his brethren to follow. Their search leads them to Scarlet
   Overkill, the world's first-ever super-villainess."
}
```


Insidious on "www.rottenmelons.com"


```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "Insidious",
  "actors": ["Patrick Wilson", "Rose Byrne", "Lin Shaye"],
  "director": "James Wan",
  "contentRating": "PG-13",
  "genre": ["Horror", "Mystery", "Thriller"],
  "productionCompany": "Alliance",
  "datePublished": "2011-07-21",
  "duration": "1H43M",
  "description": "Parents (Patrick Wilson, Rose Byrne) take drastic
   measures when it seems their new home is haunted and their
   comatose son (Ty Simpkins) is possessed by a malevolent entity."
```

```
}
```

Insidious on "www.movieFacts.at"

```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "Insidious",
  "actors": ["Patrick Wilson", "Rose Byrne", "Lin Shaye"],
  "contentRating": "PG-13",
  "genre": ["Horror", "Mystery", "Thriller"],
  "company": "Alliance",
  "datePublished": "2011-07-21",
  "description": "Parents (Patrick Wilson, Rose Byrne) take drastic
    measures when it seems their new home is haunted and their
    comatose son (Ty Simpkins) is possessed by a malevolent entity."
}
```

Spirited Away on "www.movieFacts.at" and "www.rottenmelons.com"

```
{
  "@context": "http://schema.org",
  "@type": "Movie",
  "name": "Spirited Away",
  "alternateName": "Chihiros Reise ins Zauberland",
  "actors": ["Daveigh Chase", "Suzanne Pleshette", "Jason Marsden"],
  "director": "Hayao Miyazaki",
  "contentRating": "PG",
  "genre": ["Anime", "Adventure", "Fantasy"],
  "productionCompany": "Studio Ghibli",
  "datePublished": "2003-06-19",
  "duration": "2H5M",
  "description": "10-year-old Chihiro (Daveigh Chase) moves with her
    parents to a new home in the Japanese countryside. After taking a
    wrong turn down a wooded path, Chihiro and her parents discover an
     amusement park with a stall containing an assortment of food. To
    her surprise, Chihiro's parents begin eating and then transform
    into pigs. In this supernatural realm, Chihiro encounters a host
    of characters and endures labor in a bathhouse for spirits,
    awaiting a reunion with her parents."
}
```