terkait - Semantic Interactions with Web Content

Sebastian Germesin
Deutsches Forschungszentrum für Künstliche Intelligenz
Campus D3 2
66123, Saarbrücken, Germany
sebastian.germesin@dfki.de,

Massimo Romanelli
Deutsches Forschungszentrum für Künstliche Intelligenz
Campus D3 2
66123, Saarbrücken, Germany
massimo.romanelli@dfki.de

ABSTRACT

We present our ongoing development of an application, called *terkait* - that supports a user with related information while browsing the web. *terkait* analyses semantic objects that are either annotated directly in the content or retrieved by external services. The extracted entities are then preprocessed, filtered and finally presented to the user in a non-obtrusive user interface. The user can then choose to learn more about certain knowledge objects by browsing the related content that is presented alongside the particular entity. The architecture is implemented with clean and open APIs to allow different services to be used for both, the analyzing part as well as the querying for related content.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Information browsers; H.5.3 [**Group and Organization Interfaces**]: Web-based interaction

General Terms

Design, Performance

Keywords

Semantic Web, RDFa, Related content, Semantic lifting, User interface

1. INTRODUCTION

In the late 1990s, the popularity of the world wide web (WWW) and its exponential growth made key interactions such as, e.g., *creation*, *querying* and *maintenance* of content increasingly difficult [2]. The resulting movement towards solutions that use machine-readable semantics for the content of the WWW has developed the "Semantic Web" [1]. Standards (e.g., RDF(a), Microdata, OWL), ontological representations (e.g., Schema.org[1], FOAF[2]) and databases (e.g., Linked Open Data, DBPedia) have been developed and adapted to the WWW. Based on these developments, more and more toolkits, libraries and services are realized that continuously raise the Semantic Web from its scientific roots into the real world where it is faced with industrial needs (cf. [4]).

We present here our ongoing development of an application that is built on top of existing libraries and semantic back-end services to provide an end-user with related content while browsing the web. The codename of this application is ***terkait*** which is implemented in pure Javascript making it flexible enough to allow the integration in different environments, e.g., a direct integration in web-pages, a plug-in development for browsers, a web-application for mobile devices or a widget for the dashboards on modern operating systems. Naturally the development of non-obtrusive interfaces for each environment is difficult and should be designed and implemented carefully. As the focus of this publication lies on the internal architecture and API, we refer to an implementation of *terkait* as an extension for the popular browser Google Chrome.

2. MISSION

terkait's main goal is to support a user with related information when browsing through the web. It offers a user interface to present related content based on a semantic analysis of the content from the webpage the user is currently looking at. The analysis (or *"semantic lifting"*) is mainly performed using existing web services, e.g., Apache Stanbol[3] or OpenCalais[4] but furthermore parses the content for pre-annotated knowledge using a parser for RDFa and Microdata markup.

The principle of a "browsing assistant" is similar to other applications, e.g., the Piggy Bank ([3]) or the integration of the Google Knowledge Graph into certain search results in Google. However, the key improvement with *terkait* is that it offers a clean and open-sourced API to enable developers to individualize the behavior of the application by supporting more backend services which have access to deep-web information or by extending the renderers of entity types to broaden the field of application of *terkait.*

terkait works in two different modes: In the **default mode** the system uses the complete content of the webpage to be

[1]http://schema.org
[2]http://www.foaf-project.org/
[3]http://incubator.apache.org/stanbol/
[4]http://www.zemanta.com/

automatically analyzed for relevant information to identify semantic objects with the help of back-end services. In this mode, the number of interactions a user has to perform is minimized to allow a non-obtrusive behavior. However, in rare cases, the services do not perform with 100% accuracy and might miss important information or present unimportant results to the user. In order to face such cases, we have extended *terkait* with the **power-user mode**. This mode enables experienced users to explicitly annotate elements on a web-page which *terkait* can then rely on. Naturally, the latter mode results in exactly the information the user is in need of, but requires some level of expertise to annotate certain elements.

3. ARCHITECTURE

terkait is implemented in Javascript to allow an easy integration into different environments. For the sake of simplicity, we present the system in an implementation as an extension to the Google Chrome browser. Our motivation of implementing *terkait* as Google Chrome extensions comes with the ability to publish it in the central repository of the Google Web Store[5]. However, the extension-specific code has been kept apart from *terkait*. Figure 1 shows the data flow through the system. After an initial content cleansing stage, different semantic lifting services are queried to analyze the content of the current webpage. The results are merged, filtered for duplicates and ranked by relevance. In parallel, different knowledge hubs are queried to enrich each entity with possible missing attributes. Entity-specific renderer finally present each entity along with a short abstract. Triggered by the user, *terkait* is able to query pre-defined services for entity-related content. Each of these steps will be described in detail in the following sections.

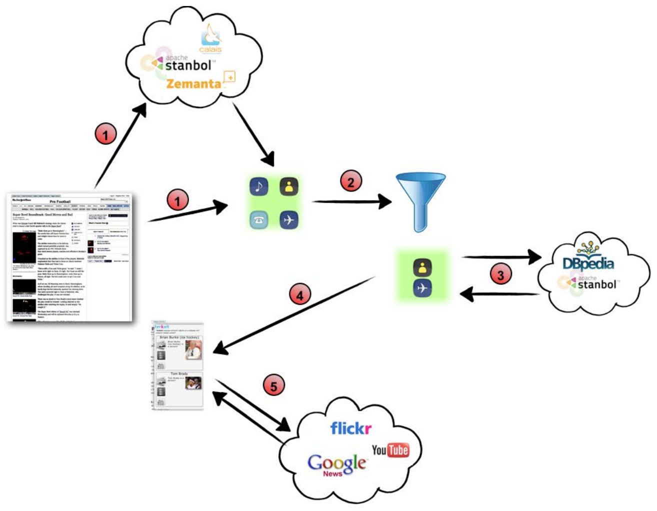


Figure 1: Data flow through *terkait*.

3.1 Content Cleansing

In order to present related content, *terkait* needs to analyze the content of the webpage with available semantic back-end services. However, sending the full page to these services not only takes a lot of bandwidth but also reduces the accuracy of these services. A pre-processing and filtering that removes obviously unnecessary information on a web-page (e.g., navigation bars or advertisements) from the *content-of-interest* is reasonable. Historically, *terkait* used a self-implemented local algorithm to filter such content. However, using APIs like Alchemy[6] or Readability[7] is a more robust approach, as these services rely on a larger set of resources and features for this task. Hence, *terkait* sends the URL of the webpage to such a service and forwards the returned text in the next step.

3.2 Semantic Lifting

Given an available broadband internet connection, it is always preferable to use dedicated services over downsized local solutions for tasks that require multiple knowledge sources and an extensive amount of computing power. Such a case is the extraction of knowledge from unstructured web content (e.g., text or images). Though basic information is already available through semantic annotations using markups and formats like, e.g., RDFa or MPEG-7, the deeper meaning of the content is usually extracted by semantic back-end services. Currently, *terkait* has implementations of connectors to the following services: Apache Stanbol, Zemanta and OpenCalais. These services allow to analyze unstructured textual-content using natural language technologies and to extract entities of types *Person*, *Place* and *Organization*. The clean API of *terkait* as well as the underlying VIE.js library (see section 3.7) make it easy to extend the number of services at any time in the future with services that may extract information from, e.g., multimedia content.

3.3 Filtering

Once the back-end services returned the semantic information in form of triples, *terkait* applies certain filters in the second step to remove duplicate entries and irrelevant entity-types. This is necessary as *terkait* might not support the presentation of all types of entities or the user might have explicitly selected a subset of entity-types to be filtered for. One problem is that the information from the back-end services and from the locally annotated content might not include enough type-specific information for the filtering. In order to prevent false-negatives in the filtering process, *terkait* uses *deliberate filtering*: Only those entities are removed that contain enough information to ensure their removal. The remaining entities that either have information that they should not be filtered or do not provide enough information are kept in the data flow. The filtering process is re-applied on these remaining entities after the enrichment phase in step three (see upcoming section) once their missing data has been acquired.

3.4 Enrichment

Especially when parsing semantic annotations from a webpage, it is unclear whether all knowledge of the referenced entities is provided or only a subset. Though, semantic back-end services usually include basic information about an entity in their results (e.g., `rdf:label`, `rdf:type` or `foaf:depiction`), most applications need more data for their processing. In step three, we enrich the entities that have been acquired in the previous step by explicitly querying semantic

[5]terkait - http://goo.gl/QLJKm

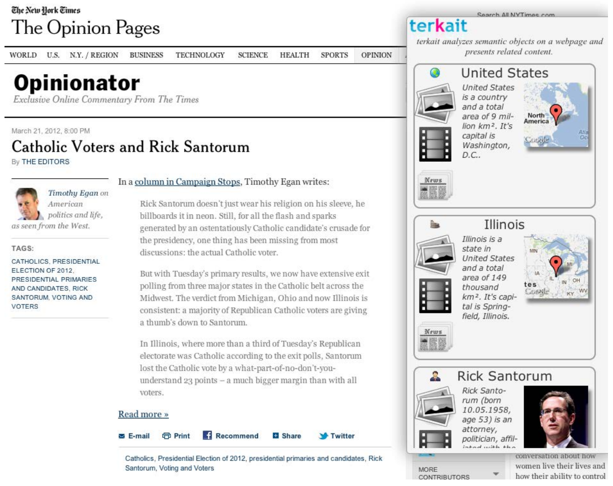
[6]http://www.alchemyapi.com/

[7]http://readability.com

databases (e.g., the entity hub from an Apache Stanbol installation or a DBPedia[8] endpoint) to fill missing attributes for the remaining entities. Again, the API of *terkait* allows to easily extend the number of semantic databases in the future.

3.5 Rendering

The previously described steps allow *terkait* to construct this context whereas the *rendering* phase presents the user with information about the context. The subsequent step finally describes how the user has access to the related content.


The New York Times
The Opinion Pages
WORLD U.S. N.Y. / REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORTS OPINION
Opinionator
Exclusive Online Commentary From The Times

March 21, 2012, 8:00 PM
Catholic Voters and Rick Santorum
By THE EDITORS

Timothy Egan on American politics and life, as seen from the West.

TAGS:
CATHOLICS, PRESIDENTIAL ELECTION OF 2012, PRESIDENTIAL PRIMARIES AND CANDIDATES, RICK SANTORUM, VOTING AND VOTERS

In a column in Campaign Stops, Timothy Egan writes:

Rick Santorum doesn't just wear his religion on his sleeve, he billboards it in neon. Still, for all the flash and sparks generated by an ostentatiously Catholic candidate's crusade for the presidency, one thing has been missing from most discussions: the actual Catholic voter.

But with Tuesday's primary results, we now have extensive exit polling from three major states in the Catholic belt across the Midwest. The verdict from Michigan, Ohio and now Illinois is consistent: a majority of Republican Catholic voters are giving a thumb's down to Santorum.

In Illinois, where more than a third of Tuesday's Republican electorate was Catholic according to the exit polls, Santorum lost the Catholic vote by a what-part-of-no-don't-you-understand 23 points – a much bigger margin than with all voters.

Read more »

E-mail Print Recommend Share Twitter

Catholics, Presidential Election of 2012, presidential primaries and candidates, Rick Santorum, Voting and Voters

terkait
terkait analyzes semantic objects on a webpage and presents related content.

United States
United States is a country and a total area of 9 million km². It's capital is Washington, D.C..

Illinois
Illinois is a state in United States and a total area of 149 thousand km². It's capital is Springfield, Illinois.

Rick Santorum
Rick Santorum (born 10.05.1958, age 53) is an attorney, politician, affiliated with the

Figure 2: Screenshot of *terkait*.

terkait's context contains semantic knowledge about entities that have been extracted from the content. These entities are ranked by the number of occurrences and relations within the context. The graphical interface of *terkait* is shown in Figure 2 where the most relevant entities (based on the calculated ranking) are rendered above the less relevant information. However, the ranking algorithm can also incorporate geo-location information to ensure that persons that live in Illinois would not be presented with information about that city - based on the presumption that they know already information about their city.

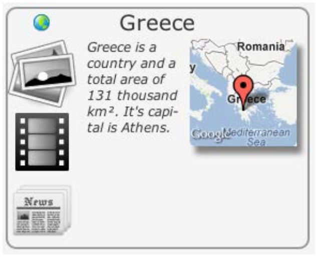

Greece
Greece is a country and a total area of 131 thousand km². It's capital is Athens.

Figure 3: Rendering of a country in *terkait*.

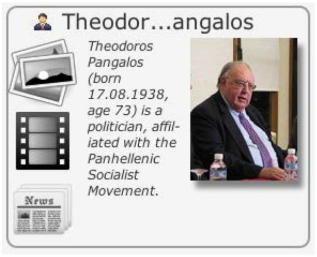

Theodor...angalos
Theodoros Pangalos (born 17.08.1938, age 73) is a politician, affiliated with the Panhellenic Socialist Movement.

Figure 4: Rendering of a politician in *terkait*.

We implemented a set of renderers to leverage the semantic information about each entity for the presentation. Though the main parts of a panel are preset and fixed in order to provide a consistent user interface, subtle differences are useful when presenting different types of entities. When presenting information about a country (see Figure 3) it makes sense to present a map with the location of that place and some information about its size. On the contrary, the renderer of an entity of type `dbpedia:Person` (see Figure 4) would present a depiction of that person alongside with information about its age, rather than presenting a map.

It is notable that all aforementioned queries and processings are performed asynchronously. While this keeps the browser responsive this fact plays a special role when it comes to the presentation of the information in the rendering step. User tests with *terkait* have shown that a fast presentation of incomplete information at an early stage is preferred over a long shot, where nothing is displayed until all information is gathered. However, once new information is available, the changes in the rendering need to be performed smoothly to avoid flickering of the user interface.

3.6 Related Content

With the help of *terkait*, the user can explicitly ask for related multi-medial content, e.g., images, videos or news articles, that are related to the entity in focus. This is of particular interest for the user as it broadens the user experience of the web as it is now. Currently, only predefined links can be followed and provide more information about the current content. By automatically analyzing the content and adding semantics, that knowledge is leveraged in search queries to allow to browse information beyond what is shown on the webpage. Figure 5 is a screenshot of the user interface after the user has clicked on the *images* button to trigger the query for images that are related to the entity *"United States"*.

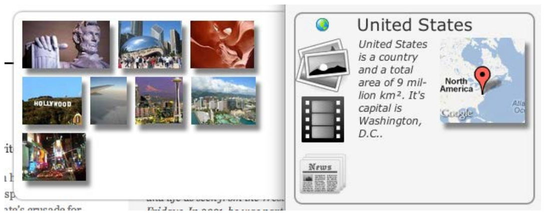

United States
United States is a country and a total area of 9 million km². It's capital is Washington, D.C..

Figure 5: Screenshot of related images for a country.

The architecture in *terkait* is kept open to allow developers to add other content to be queried or to replace an existing implementation for, e.g., images with its own implementation that relies on a different service. Each query only depends on the service itself and the entity-in-focus. Most services are non-semantic services and hence *terkait*'s implementation allows to transform the concept "Paris Hilton" into a specialized query that retrieves images of the celebrity rather than the hotel in the city in France. Note that such a combination and transformation is implemented in Javascript code that needs to be written by a developer.

3.7 VIE.js - Semantic Interactions

terkait is based on the VIE.js[9] library. This library is written in Javascript and offers a domain-specific language

[8]http://dbpedia.org

[9]VIE - Vienna IKS Editables - http://viejs.org

to ease recurring tasks when dealing with semantic entities. In *terkait*, this library is especially used to query the semantic back-end services and to parse the semantic annotations. Furthermore, VIE.js easily allows to filter entities based on their type hierarchy and - as it is based on the well-known MVC library Backbone.js[10] - it offers an easy-to-use API to request and update relations of entities. Moreover, the VIE.js developers recently started to offer a user interface widget library. Some of the available widgets have been used directly in *terkait*, e.g., the querying for related content is implemented with the help of the VIE.js content-retrieval widget. The benefit of re-using that library in the implementation is to allow other developers to easily extend *terkait* for their purposes.

3.8 Evaluation

During the development of *terkait*, the software has been presented internally in our research lab to small groups to gather first feedback. Naturally, these audiences do not cover the targeted personages of users but were already helpful in cleaning the user interface to, e.g., improve the navigation to related content.

Moreover, more than 50 installations have been registered at the Google Chrome Web Store and incoming requests for features and issues with *terkait* will help to develop it further in future releases.

However, a more thorough user study is planned where users[11] are given a browser with an installed version of *terkait*. The users are given a task of 30 minutes to navigate through (pre-chosen) news articles and actively use *terkait* to find related information about (pre-defined) terms / entities. After each webpage, the user is given a questionnaire to rate the performance of *terkait* . Table 1 shows a subset of questions from that questionnaire.

relevance	*"Please rate the ranking of the entities on a scale from 1(very bad) to 5(very good)". "Please rate the relevance of the related content on a scale from 1 to 5". "Please rate the relevance of the shown description on a scale from 1 to 5".*
completeness	*"Please list a number of terms/entities that you missed in the list".*
interface	*"Please rank the stacked presentation on a scale from 1 to 5". "Do you prefer a fast-response of the UI over a correct ranking or not?".*
misc.	*"Please describe features or add a comments that you have".*

Table 1: Subset of questions that are presented during the user study.

4. FUTURE WORK

We are currently extending *terkait* in different parts of the architecture. In the current implementation, the annotations (manual and automatic) are stored as tags in *terkait*'s internal triple store in relation with the page they have been extracted from. However, a more precise information that would also include the exact location in the page would be desired and will be implemented in a future release. Furthermore this information would then also be used to present related web-pages to the user from the local browser history, based on the extracted knowledge. It is also planned to analyze the user's interaction with the presented content and entities to influence the ranking algorithm in further queries.[12]

In the upcoming release of *terkait* it will provide the user with the capability to browse relations between entities in a connected way. In figure 5 the entity *"United States"* refers to an other entity *"Washington, D.C."* but does not allow to query information about that city. That entity will be interactive and when the user clicks on it, *terkait* navigates the user to the visualization of that entity.

5. CONCLUSION

We have presented *terkait* - a browser assistant that analyses semantic objects on a web-page and offers related content to the user while browsing the web. This application is implemented in Javascript and based on the existing VIE.js toolkit. *terkait* uses two main knowledge sources to gain knowledge about the content on the web-page: On-page parsing of semantically annotated content and querying external services to semantically lift the unannotated parts of the content. All results are then merged, filtered by the user's preference, ranked by a relevance algorithm and finally presented to the user in a non-obtrusive user interface. The user can then choose to learn more about certain knowledge objects by browsing the related content that is presented alongside the particular entity. The architecture is implemented with clean and open APIs to allow different services to be used for both, the analyzing part as well as the querying for related content.

6. ACKNOWLEDGMENTS

This work is supported by the European Union's 7th Framework Programme under grant agreement FP7-ICT-2007-3/ No. 231527 (IKS). This publication only reflects the authors views and funding agencies are not liable for any use that may be made of the information contained herein.

7. REFERENCES

[1] T. Berners-Lee. *Weaving the Web.* Orion Business, 1999.

[2] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors. *Spinning the Semantic Web.* MIT Press, 2003.

[3] D. Huynha, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1), 2011.

[4] W. Maass and T. Kowatsch. *Semantic Technologies in Content Management Systems: Trends, Applications and Evaluations.* Springer Verlag, 2011.

[10] http://backbonejs.org

[11] The user study will be conducted with students from our university with a focus on diversity through the different faculties.

[12] This targets the use-case where a user constantly ignores and removes found entities from the list which - over time - will receive a lower score.