SUMMA: A Common API for Linked Data Entity Summaries

Andreas Thalhammer(✉) and Steffen Stadtmüller

Karlsruhe Institute of Technology, Karlsruhe, Germany
{andreas.thalhammer,steffen.stadtmueller}@kit.edu

Abstract. Linked Data knowledge sources such as DBpedia, Freebase, and Wikidata currently offer large amounts of factual data. As the amount of information that can be grasped by users is limited, data summaries are needed. If a summary relates to a specific entity we refer to it as entity summarization. Unfortunately, in many settings, the summaries of entities are tightly bound to user interfaces. This practice poses problems for efficient and objective comparison and evaluation.

In this paper we focus on the question of how to make summaries exchangeable between multiple interfaces and multiple summarization services in order to facilitate evaluation and testing. We introduce SUMMA, an API definition that enables to decouple generation and presentation of summaries. It enables multiple consumers to retrieve summaries from multiple providers in a unified and lightweight way.

Keywords: Web APIs · Entity summarization · Evaluation · Testing · User interfaces · Linked data

1 Introduction

With the growth of Open Data on the Web a plethora of information sources covering diverse topics and domains are readily available to information consumers. The abundance of information that can be found on single entities can even be increased with the integration of different data sources with semantic links (Linked Open Data). However, when it comes to the presentation of such information, there are often limits on the amount of data that can be rendered in interfaces and grasped by end users. These limits give rise to the requirement to only show the most important data in visualizations, i. e. a summary. The amount of commercial systems that offer entity summaries are on the rise [7,10,17]. Due to their proprietary nature, these systems tightly couple their user interface and backend in accordance to their specific requirements. Also the data sources from which these commercial summaries are derived are mostly not publicly available. As a consequence, it becomes hard to exchange, evaluate, and compare the output of summarization systems in an objective manner. In order to facilitate accessibility of entity summaries it is necessary to identify the principal properties of entity summarization systems, create a corresponding data model, and to adhere to the best practices of Web APIs.

P. Cimiano et al. (Eds.): ICWE 2015, LNCS 9114, pp. 430–446, 2015.
DOI: 10.1007/978-3-319-19890-3_28

To enable clients to easily consume the summaries of entities from different summarization services we propose SUMMA, a uniform lightweight interface design based on a request/response vocabulary and the Representational State Transfer (REST) interaction paradigm. The approach enables to combine a diverse selection of summarization approaches on a single Web site and to switch from one service to another even during user navigation. The proposed API aligns with the Linked Data interaction model. Our approach treats the summarization approach itself as a black box while preserving the possibility to define the required parameters of an entity summarization system in a uniform manner. Thus clients can easily substitute or combine the employed entity summarization system in a plug-and-play fashion. Existing summarization systems can easily offer summaries with the SUMMA API in addition to their deployed user interfaces. In order to facilitate evaluation, the API aims at supporting researchers and practitioners in the following settings:

- *Quantitative Evaluation*: SUMMA enables consumers to retrieve summaries of entities in their most pure form (a ranked list of RDF statements). As such, reverse engineering tasks such as disambiguating strings to URIs (e.g. mapping "`recompense`" to http://dbpedia.org/ontology/award) are not needed for automatic comparison of different approaches.
- *Qualitative Evaluation*: Commonly, multiple systems are placed next to each other in qualitative evaluation settings for entity summarization and subjects are asked to choose one or more. To support this, a SUMMA client can present summaries of multiple different summarization systems in a uniform way. In this way it can be ensured that style elements (such as pictures, borders, colors, etc.) do not play a significant role in the subjects' decision making process.
- *A/B Testing*: Evaluation with A/B testing is commonly applied in industry settings. SUMMA enables to change the technology that produces summaries while the user interface stays the same. By tracking the interaction with each variant it is possible to compare the effects of technology changes.

For our approach, we provide an open-source reference implementation and deployment as well as an empirical evaluation. The source code of the reference implementation as well as a deployment are available online. In the empirical evaluation, we measure the overlap of our established requirements with the features of real-world systems. This study includes interfaces of well known search engines like Bing, Google, and Yahoo as well as entity presentations of well-known news portals.

The remainder of this paper is organized as follows: In Section 2 we present a requirement analysis for a uniform entity summarization API as well as the API itself. In Section 3 we introduce the implementation as well as its deployment. The evaluation in Section 4 introduces an empirical study that assesses the applicability of our approach to real-world user interfaces. In Section 5 we analyze the most related approaches and outline how we differ from them. Section 6 concludes the paper and provides an outlook on our future work.

2 SUMMA API Definition

In its most basic form, a summary of an entity can be produced by two given parameters:

URI A URI that identifies the entity.
k A number k that defines an upper limit of how many facts about the entity should be presented.

While it is obvious that there is a need for an unambiguous reference to the entity, it could be argued that a summary could also be specified by a given compression level. For example, we could specify that 30 % of all facts about the given entity should be contained in the summary. In this respect, we would like to point out that concise presentations (for which we are aiming) are better declared with an upper limit rather than a given percentage. This is due to the fact that knowledge bases commonly cover well documented entities as well as a long tail of sparsely documented ones: in this respect, 30 % could mean 20,000 facts for some entities while only 3 for others.

When defining a uniform interface for entity summarization, various specifics that are inherent to the definition of RDF itself have to be considered as well. This ranges from the possibility to have multiple labels for vocabulary or data items to the more complex summaries that consider n-ary relations[1] or enable full property chains. Next to these features, other requirements include the grouping of statements and the restriction to a predefined set of properties. In the following we present an overview of all further requirements of the API:

Languages In many knowledge bases, labels in different languages for resources and properties are commonly available. In order to avoid multiple requests or queries to different knowledge sources we find it necessary to include labels of one or more languages in the output of the summary.
Multi-hop Search Space It might be necessary (think of n-ary relations or reification) or interesting to include statements in the summary that do not directly involve the targeted entity but are connected through one or more hops. For example, a max hop parameter of 1 (default) only considers statements where the entity is either in the subject or object role, while a max hop of 2 could cover facts that are still about the entity but are modeled via an n-ary relation. Further hops are possible.
Property Restriction A summary can be targeted to a predefined set of properties. An example would be to restrict the summary of a movie to {`dbpedia:starring`} or {`dbpedia:starring, dbpedia:director`}. This feature is very useful if the interface has reserved space for specific features such as a map presenting geolocations or pictures. These features can be retrieved in a separate request.

[1] "Defining N-ary Relations on the Semantic Web" – http://www.w3.org/TR/swbp-n-aryRelations

Statement Groups Rather than ranking statements only individually, the system could form groups or clusters of statements and, if applicable, provide names for these groups.

These features and their compositions enable very specific views on entities although they are still abstract enough to be applicable to any knowledge base, be it encyclopedic or proprietary. In general, also the following considerations have to be taken into account:

Resources/Literals Linking to other resources (i.e., URI identified entities) supports exploration aspects while textual information (represented as literals) satisfies more the information need about the specific entity. For visualization purposes any resource URI included in a summary has to be accompanied by a literal description which enables a user-friendly rendering of the resource. Clients consuming the summary can therefore ignore the resource URIs and only use literals for presentation.

Outgoing/Incoming Links For any unidirectional relation `:x :link :y` a second relation can be established in the way `:y :link_by :x`. In many cases displaying such a relation in a summary of `:y` makes sense as it covers information about it. Knowledge bases such as DBpedia, Freebase, and Wikidata enable to retrieve incoming links from other resources of the respective knowledge base with queries. For Linked Data in general, many incoming links can be retrieved with crawls as provided e.g. by the BTC [5].

Our approach consists of two main components with a strong interplay:

- The **SUMMA Vocabulary** can be used to frame summary requests, which can be submitted to a summarization engine. Servers can interpret the given parameters in the request and produce result sets with the vocabulary that are in accordance to the provided parameters.
- The description of the **RESTful Web Service** provides a clear guideline for the interplay between summary consumers and producers.

In the following, we first introduce the *SUMMA Vocabulary* and thereafter the *RESTful Web Service* interaction guideline.

2.1 SUMMA Vocabulary

The *SUMMA Vocabulary* offers various parameters that help to configure and represent a summary. During the design of the vocabulary we took the above considerations into account. An overview of the vocabulary is depicted in Figure 1. In the following we introduce all classes and properties:

Summary This class describes the abstract concept of a summary of an entity. The URIs of instances of this class are constructed with all query parameters.

SummaryGroup This class describes a group of statements. The entity summarization system does not necessarily have to produce groups. If groups are formed, it is completely up to the summarization system what is meant by them or if they come with a label in the desired language.

entity This predicate with domain Summary and range `rdfs:Resource` points to the entity that is summarized. As an example, the object of this property could be a DBpedia or Freebase entity. This property is mandatory for the API.

topK This property defines the maximum number of statements that are being returned. This property is mandatory for the API.

statement This property with domain Summary and range `rdf:Statement` attaches statements to a summary in the response context.

maxHops This property defines the maximum number of hops in the graph the interface is able to represent. The default value is set to 1, which means that all properties in the immediate vicinity of the focused entity are being considered.

path This property enables to include the full paths in the returned statements of the summary. For each statement that is included in the summary that does not directly involve the focused entity, a path that shows how the current statement relates to the entity needs to be provided. This situation can occur if the maximum of hops is greater than 1. For more than 2 hops, this relation is needed multiple times until the object statement of path includes a triple that contains the focused entity.

language This property defines the languages in which the output literals should be available. We recommend to use a fixed vocabulary like RFC 4646[2] for this.

group The group property enables summaries to form groups of statements. Attaching a group directly to a statement enables clients to ignore the property if present but not supported.

fixedProperty If there is already some background knowledge on the summarizer's side about the underlying data structure it can request properties that it wants to show in any case. Multiple different properties can be defined in this way and thereby restricting the output to the defined set of properties.

Next to this vocabulary, we make use of the vRank[3] vocabulary [9], XSD[4] and OWL[5]. The vRank vocabulary is necessary to include the computed scores of each statement by the summarization service. A summary typically includes more than one `rdf:Statement`. Although in some syntaxes constructs such as `summa:statement [ a rdf:Statement; ...], [ a rdf:Statement; ... ] .` could be mistaken for ordered lists, the group of statements is returned as a set. To determine an order between the statements additional information is required. In this respect, we choose to use vRank rather than `rdf:List` to enable summarization systems to publish the ranking scores. Listing 1.1 exemplifies the use of the vRank vocabulary in combination with a reified `rdf:Statement`.

[2] RFC 4646 – http://www.ietf.org/rfc/rfc4646.txt
[3] vRank – http://purl.org/voc/vrank#
[4] XSD – http://www.w3.org/2001/XMLSchema#
[5] OWL – http://www.w3.org/2002/07/owl#

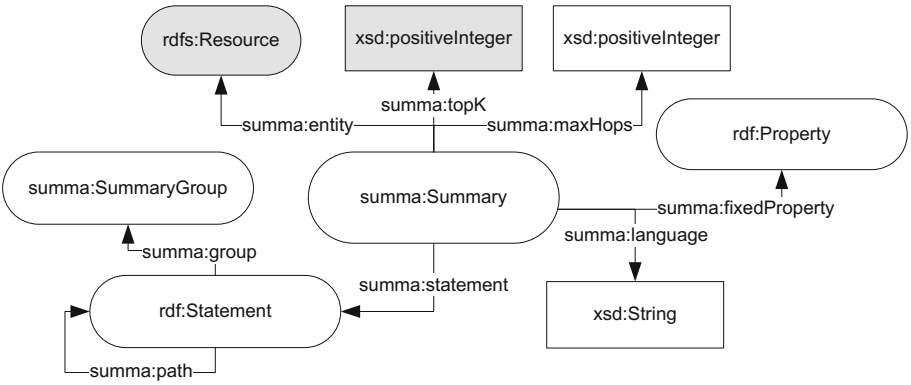

Fig. 1. The SUMMA Vocabulary. Mandatory parameters in grey

The *SUMMA Vocabulary* is published at http://purl.org/voc/summa/. Exemplary usages of the vocabulary are shown in Listing 1.2 for input and in Listing 1.3 for output (see Appendix A).

Listing 1.1. Example for using vRank for ranking a RDF Statement.

```
1
2
3  [ rdf:type rdf:Statement ;
4    rdf:subject dbpedia:Barack_Obama ;
5    rdf:predicate dbpedia-owl:birthDate ;
6    rdf:object  "1961-08-04"^^xsd:date ;
7    vrank:hasRank [ vrank:rankValue "33.11"^^xsd:float] ]
```

Listing 1.2. Example for a summary request that is sent via POST (namespaces omitted).

```
1  [ a :Summary ;
2    :entity dbpedia:Barack_Obama ;
3    :topK "2"^^xsd:positiveInteger ;
4    :language "en" ;
5    :maxHops "2"^^xsd:positiveInteger ;
6    :fixedProperty dbpedia-owl:birthDate ;
7    :fixedProperty dbpedia-owl:birthPlace . ]
```

2.2 RESTful Web Service

According to the Richardson maturity model [8] REST is identified as the interaction between a client and a server based on three principles:

- The use of URI-identified entities.
- The use of a constrained set of operations, i. e., the HTTP methods, to access and manipulate entity representations.

– The application of hypermedia controls, i. e., the data representing an entity contains links to other entities. Links allow a client to navigate from one entity to another during his interaction.

The use of URI-identified entities and their interlinkage are also direct consequences from the Linked Data design principles[6]. Therefore, several existing approaches recognize the value of combining RESTful services and Linked Data [2,6,11,12,18,19], which led recently to the establishment of the *Linked Data Platform*[7] W3C working group.

We adopt these notions for our approach to enable a uniform interface to summarized entities that aligns with the standard Linked Data interaction model. The interaction of a client to retrieve the summary of an entity according to our approach is depicted in Figure 2 and works as follows:

1. A client can send a summary request for an entity to a server offering a summarization service via an HTTP POST request.
2. The response to the request contains the summarized entity in its payload, as well as a URI in the location header field that identifies the created summary.
3. The client can use the URI of the summary for further lookups of the summary via HTTP GET.

Since summaries can be looked up via HTTP GET, we enable simple caching mechanisms for the clients. The URI of the summary also enables to include direct links in other web resources to the summary. To construct the URI that identifies a given summary, we adopt the approach from [11], where the URI contains key/value pairs that correspond to the properties in the original summary request. Note, that the server does not have to store the created summaries for allowing the direct lookup but can calculate the summary on-the-fly for GET requests as well as by interpreting the key/value pairs in the URI.

A client can also skip the first interaction via POST and anticipate how the URI of a summary would look like as the lookups are computed in the same way as the original POST request. However, we keep both interaction schemes in place in order to enable a clear formulation of a request as well as a clean cacheable lookup.

3 Implementation

The SUMMA API definition is based on Web standards such as the HTTP protocol and RDF. Summary producers as well as consumers can be implemented in a variety of programming languages. However, in order to demonstrate feasibility and to facilitate adoption, we provide a reference implementation based on Java Jersey[8] (server) and JavaScript (client).

[6] http://www.w3.org/DesignIssues/LinkedData.html
[7] http://www.w3.org/2012/ldp/charter
[8] Jersey – https://jersey.java.net/

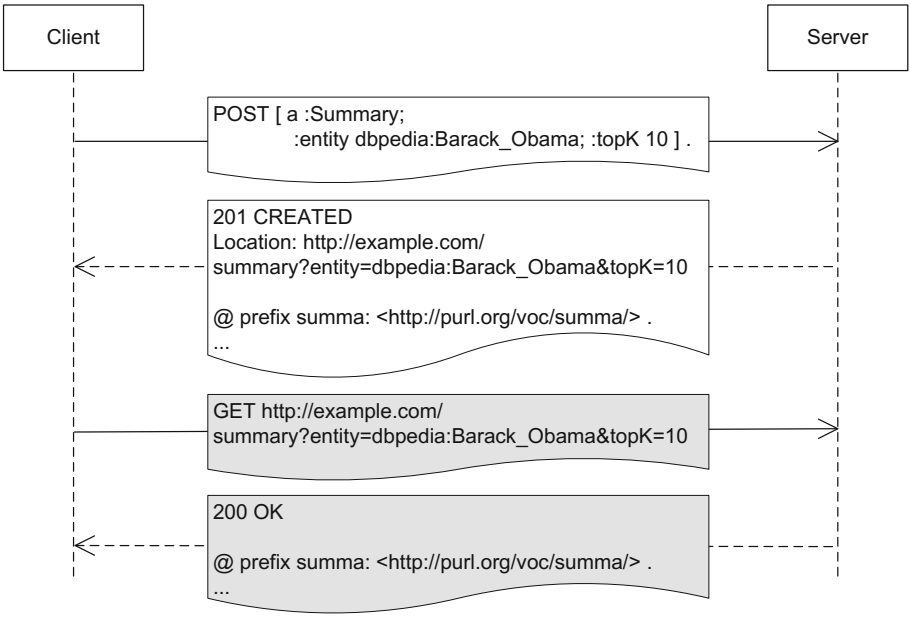

Fig. 2. Messages for first interaction: white. Messages for second interaction: grey

summaServer The summaServer application is an Apache Tomcat server application that fully implements the SUMMA API. It provides a naïve summarization method for DBpedia entities. This summarization method ranks objects (only outgoing links are considered) based on the number of their incoming links within Wikipedia. All necessary information (including the link counts) is available via the official DBpedia SPARQL endpoint.[9] The source code of summaServer application is published at https://github.com/athalhammer/summaServer and licensed with GPLv3.[10] Deployments of the summaServer application and another summarization method [15] can be found at the following addresses:

- http://km.aifb.kit.edu/summaServer
- http://km.aifb.kit.edu/summa

summaClient The summaClient library is a lightweight JavaScript application that interacts with servers that implement the SUMMA API. It enables visualization and interaction with the results of multiple summarization engines at a single Web page (see Figure 3). The source code of the summaClient library is published at https://github.com/athalhammer/summaClient and licensed with GPLv3. A deployment of the summaClient library can be found at http://people.aifb.kit.edu/ath/summaClient/.

[9] DBpedia SPARQL endpoint – http://dbpedia.org/sparql
[10] GPLv3 – http://www.gnu.org/copyleft/gpl.html

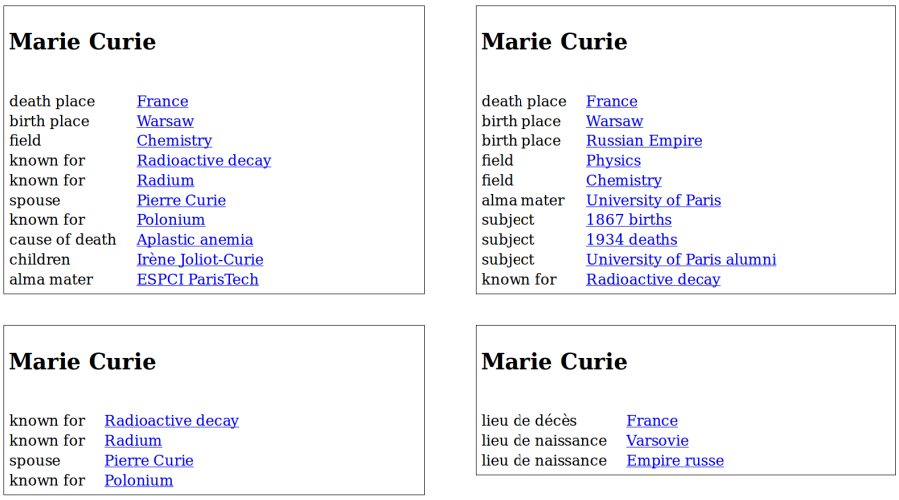

Fig. 3. Screenshot: Two example summaries with the same configuration but different systems (top). Example summary with restriction to two properties (bottom left) and a different language and $topK = 3$ (bottom right).

4 Evaluation

In our evaluation we inspect interfaces from well-known providers such as the Google Knowledge Graph (GKG) [10], Microsoft Bing Satori/Snapshots [7], or Yahoo Knowledge [17]. We assess whether the expressibility of these interfaces could be served via the SUMMA API. Thus, we provide empirical evidence about the general applicability of the API for any kind of RDF entity summary.

For our evaluation, we select the entity summarization systems of the three major search engines (mentioned above) as well as systems from the Alexa Top News sites[11] that offer factual knowledge about entities. We select two of the top 25 news portals offering infoboxes about entities. These are Forbes[12] and BBC news[13]. Our hypothesis is that the defined API could serve all of these interfaces, thus potentially enabling them to switch between different entity summarization services without changing their layout. For this, we focus on five entities from diverse domains: Spain, Dirk Nowitzki, Ramones, SAP, Inglourious Basterds. These entities are representatives for a country, a person (or athlete), a band, a company (or organization), and a movie. We have to note that, at the time of writing, BBC only supports summaries of countries, Forbes supports only summaries of persons and organizations, and Yahoo only supports persons and movies. For these systems our insights will be focused on the supported types. Some of the analyzed systems use also fixed schema patterns or a combination of

[11] http://www.alexa.com/topsites/category/Top/News
[12] Forbes, e. g. http://www.forbes.com/profile/dirk-nowitzki/
[13] BBC news, e. g. http://www.bbc.com/news/world-europe-17944958

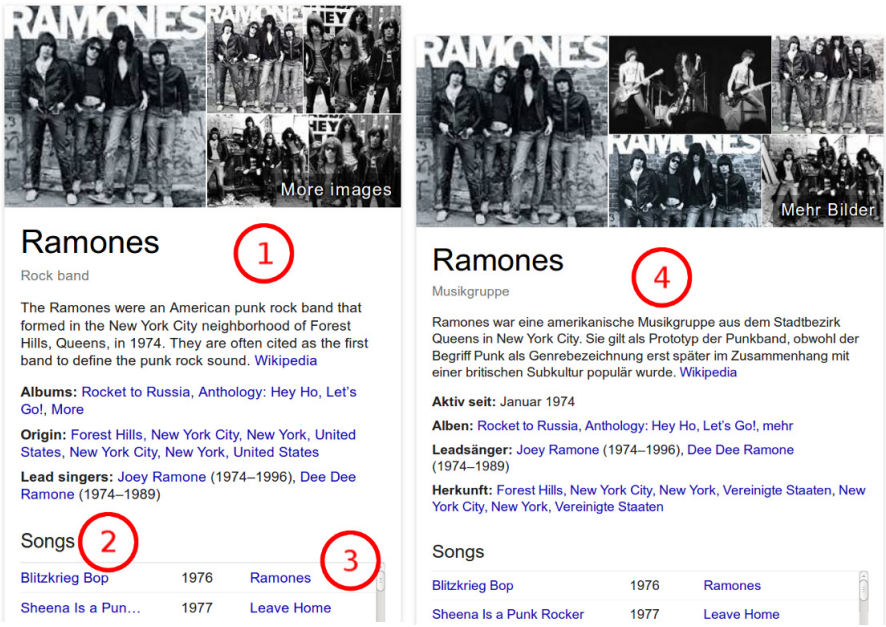

Fig. 4. Screenshot of the GKG representation of the "Ramones": 1) Specific properties such as the type and the Wikipedia description are always there (Property Restriction). 2) Several statements are gathered in a group named "Songs" (Statement Groups). 3) N-ary relations - in this case title, year, and album - are supported (Multi-hop Search Space). 4) The summary is offered in multiple languages (Languages).

entity-specific summaries and schema patterns. We assume that even with using only fixed schema patterns the content requirements at the interface remain the same. This still suits our evaluation scenario as our main goal is to decouple summary and presentation: the way in which the summaries were generated is not relevant (black box). We also tried to include research prototypes into our evaluation. However, although this research field is very active recently [1,13, 14,16,20], our system SUMMARUM [15] is currently the only research prototype that is available online.

In the following we will analyze for each of the above-mentioned interfaces on whether they would be able to consume data from the API without changing their layout. We assume both, the URI of the entity and the maximum number of facts (topK) as standard parameters. Figure 4 demonstrates the analysis of the interfaces.

Google Knowledge Graph For some facts, GKG uses contexts about the data items (e. g. Wikipedia abstracts, population numbers, dates of marriage, release year of album, role names, etc.). In RDF, these contexts are represented as n-ary relations. Our API supports summaries over such constructs with the multi-hop search space. Further, certain properties such as

entity names, pictures, or types are always present in GKG. Not considering the result of the dynamic ranking, these properties can be addressed with a separate summary request that involves fixed properties. Further, GKG supports special groups of statements, such as a the group of albums of a band. We support this feature by enabling to add a group to each statement by the entity summarization system. GKG is able to adapt the interface to different languages. This is supported by RDF (multi-linguality of `rdfs:label`, i. e. literals) and by a parameter for the entity summarization system.

Bing Satori/Snapshots Bing Snapshots also supports features similar to the GKG (i. e. context, special property selection, grouping, multiple languages). Bing enables tables like "Career" vs. "Season" statistics in their summaries. Even these statistics can be broken down to triples and represented in our output format. How the triples are arranged in the end, in a table style or just sequential is a matter of choice on the interface side. Certain patterns in the output (e. g. multiple numerical values with the same property but varying context) suggest table-style presentation.

Yahoo Knowledge At the time of writing, Yahoo displays factual knowledge about persons and movies. The output for movies is very similar to the aforementioned summarization systems of Google and Microsoft. Similar to Bing, the output for Dirk Nowitzki includes various sport statistics. Like in Bing, this data can be covered by our output model. Entities representing other persons are very similar to the standard output of Google and Bing. Yahoo currently does not offer summaries in multiple languages.

Forbes The interface shows basic properties of persons and companies in a key-value style. Selected properties such as the label or a picture are present for any entity. Similar to GKG, for some properties the context is added, e. g. "As of June 2014". For companies, Forbes forms two groups. "At a Glace" and "Forbes Lists". All these features are supported by our defined data model. Like Yahoo, Forbes does not offer their summaries in different language versions.

BBC news The BBC news portal includes summaries of countries only. Like in Forbes, this data contains mainly key-value pairs and is easy to be represented with our output format. Also presenting multi-hop information is needed, as the presented images have a caption that is also shown. BBC does not define groups of facts and does not offer other languages than English.

The complete results of the evaluation are presented in Table 1. Overall we found that all the requirements that these interfaces need in order to offer all their functionality can be fulfilled by the proposed API.

5 Related Work

For the related work we distinguish between two kinds of approaches: systems that add an additional layer between a SPARQL endpoint and data consumers (as such serving as direct data providers) and approaches that introduce formalisms that enable ranked views on Linked Data.

Table 1. Requirements per interface. The checked features are supported by the specific interface, the crossed ones are not required.

Features	Google	Bing	Yahoo	Forbes	BBC
Languages	✓	✓	×	×	×
Multi-hop Search Space	✓	✓	✓	✓	✓
Property Restriction	✓	✓	✓	✓	✓
Statement Groups	✓	✓	✓	✓	×

Pubby[14] is used to add an intuitive interface to SPARQL endpoints. It enables to consume entities and ontologies on a per-concept basis directly in various formats. For entities, it considers attached literal values in all available languages as well as all incoming and outgoing relations. In general, Pubby implements the following pattern for resources described by their `URI`:
`SELECT * WHERE {{<URI> ?p ?v . } UNION { ?v ?p <URI> . }}`
This may result in a large set of facts that are directly related to the currently browsed entity. For machines as well as for human consumers all information about an entity is provided. In our approach we extend this mechanism by various configurable properties (e. g. maximum number of statements) that enable client interfaces to retrieve distilled versions of entities in a uniform way.

The Linked Data API[15] adds a RESTful layer on SPARQL endpoints. It enables developers who are not familiar with SPARQL or RDF in general to access SPARQL endpoints in a RESTful manner. As an example, it enables to represent selectors and filter options as request parameters in the following form: `http://example.com/university?country=UK&max-noStudents=10000`.
Potential response formats include JSON, XML, RDF/XML, and Turtle. The Elda[16] system provides a reference implementation for the Linked Data API definition. The Linked Data API and SUMMA both add an additional RESTful layer on top of SPARQL endpoints. However, the rationales of both approaches are complementary: while the Linked Data API tries to make part of the SPARQL feature set more intuitively accessible using REST, we are focusing on defining a uniform RESTful interface that enables multiple services to provide concise views on the same entity in a uniform way.

Bizer et al. define Fresnel,[17] a vocabulary for selecting and formatting RDF data [3]. The vocabulary is supported by RDF browsers such as Longwell[18], Piggy Bank[19], or IsaViz[20]. It is devided into two main components, lenses and formats. While the lenses help on selecting which content should be presented the formats define the style in which the selected content should be presented.

[14] Pubby – http://wifo5-03.informatik.uni-mannheim.de/pubby/
[15] Linked Data API – https://code.google.com/p/linked-data-api/
[16] Elda – https://github.com/epimorphics/elda
[17] Fresnel – http://www.w3.org/2005/04/fresnel-info/manual/
[18] Longwell – http://simile.mit.edu/wiki/Longwell
[19] Piggy Bank – http://simile.mit.edu/wiki/Piggy_Bank
[20] IsaViz – http://www.w3.org/2001/r1/IsaViz/

Our work is mostly related to Fresnel Lenses: The predicates `fresnel:instanceLenDomain` and `fresnel:classLensDomain` define the levels on which the lenses can be applied. The predicates `fresnel:showProperties` and `fresnel:hideProperties` define which properties of the instance or class are commonly shown and in which order. The order is defined with `rdf:List`. Moreover, the Fresnel Selector Language (FSL)[21] enables to define further restrictions, for example which properties of connected entities should be shown (e. g., `foaf:name`). The `fresnel:instanceLensDomain` in combination with the `fresnel:showProperties` predicate and FSL enable quite particular decisions on which triples are included in the output and which are not. Eventually, however, covering specific triples for the output with Fresnel involves complex FSL patterns and, more importantly, still only provides a description of which information should be presented but not the information itself. Summarizing entities with respect to their individual particularities is possible but the lens descriptions would already cover much of the actual data. The remaining information such as the objects and all labels would have to be gathered at a different place. In other words, SUMMA provides access to entity-specific data while Fresnel, more abstractly, was designed to operate on the class level and to provide views. In fact, there are efforts to identify the most common properties per DBpedia class with surveys and crowd sourcing and to publish them as Fresnel lenses [1]. The SUMMA API could be used for interpreting such class-level lenses and for delivering the respective content accordingly. In addition, the SUMMA API explicitly enables entity-specific summaries that are beyond the scope of Fresnel.

Federated SPARQL queries[22] offer the possibility to query knowledge bases distributed over multiple endpoints with a single query. Summaries that are computed offline could be stored at one endpoint while the actual summarized knowledge base that contains further information (such as labels) is available at a different endpoint. A single federated query would retrieve triples specific to an entity while the SPARQL `LIMIT` clause would enable different summary sizes. As in our approach, the endpoint for the summary can be easily exchanged. Summaries that are computed online (e. g., depending on the user's geolocation, language, the time of the day, etc.) can get too complex in order to be retrieved with SPARQL queries of any kind. Intermediately storing the result in an endpoint in order to make it retrievable with SPARQL adds significant overhead to a process that needs to be performed in a range of few 100 milliseconds.

Roa-Valverde et al. introduce a vocabulary for sharing ranking computations over RDF data [9]. This enables to provide detailed information about ranking computations in RDF. Properties include ranking values and time stamps as well as algorithm descriptions and configurations. We use the vRank vocabulary in order to provide ranking values to the client interface.

Harth introduces VisiNav [4], a system that allows for new interaction principles within the Web of Data. The system is based on four key concepts that support search and navigation: *Keyword Search*, *Object Focus*, *Path Traversal*,

[21] Fresnel Selector Langauge (FSL) – http://www.w3.org/2005/04/fresnel-info/fsl/
[22] SPARQL 1.1 Federated Query – http://www.w3.org/TR/sparql11-federated-query/

and *Facet Selection*. Our API clearly supports *Object Focus* as it is specifically designed to deliver entity-specific summaries. We also support *Path Traversal* and *Facet Selection*. However, the two concepts become quite similar if you do not distinguish between incoming and outgoing connections. More specifically, we slightly reinterpret the *Facet Selection* concept as we form the union rather than the intersection ("... the user can reformulate the query and obtain increasingly specific result sets" [4]). Like our approach VisiNav also provides ranked views on data. VisiNav strongly couples the user interface and the back end. As such, the rankings and views on the data can only be displayed with the VisiNav system. In this paper, we provide a way to enable decoupling of the interfaces and their respective ranking back end.

In conclusion, we can state that the idea of browsing Linked Data with concise presentations is well established and real-world applications are taking up this idea [7,10,17]. To the best of our knowledge, all previous research approaches for presenting RDF data in a concise way are based on schema patterns and do not provide the data itself. In this paper we introduce a novel approach that supports the evaluation, exchange, and comparison of entity summaries in a lightweight way.

6 Conclusion and Future Work

We introduced an API that enables entity summarization systems to publish summaries in a uniform way. Further, it enables consumers to access summaries of Linked Data entities from a multitude of summarization services through a single lookup mechanism. Our empirical evaluation shows that the SUMMA API could be applied to already existing commercial systems while the reference implementations provide evidence for feasibility and facilitate adoption. The SUMMA API for comparison and evaluation is already deployed for use in industry and research.

We are currently in the process of implementing SUMMA adapters to other summarization systems. In addition, we plan to implement a portal where different entity summarization services are gathered and described also in accordance to their non-functional properties, e. g. response time and availability. Also, context-specific and personalized summaries that lead to the extensions of the SUMMA API are currently ongoing work.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 611346 and by the German Federal Ministry of Education and Research (BMBF) within the Software Campus project "SumOn" (grant no. 01IS12051).

A Appendix

Listing 1.3. Example response in Turtle (common namespaces omitted).

```
@prefix : <http://purl.org/voc/summa/>.
@prefix vrank: <http://purl.org/voc/vrank#>.
@prefix dbpedia: <http://dbpedia.org/resource/>.
@prefix dbpedia-owl: <http://dbpedia.org/ontology/>.

<http://ex.com/summary?
entity=dbpedia:Barack_Obama&topK=2&language=en&maxHops=2&
fixedProperty=dbpedia:birthDate,dbpedia:birthPlace>
  a :Summary ;
:entity dbpedia:Barack_Obama ;
:topK "2"^^xsd:Integer ;
:language "en" ;
:maxHops "2"^^xsd:Integer ;
:fixedProperty dbpedia-owl:birthDate ;
:fixedProperty dbpedia-owl:birthPlace ;
:statement

[ rdf:type rdf:Statement ;
  rdf:subject dbpedia:Barack_Obama ;
  rdf:predicate dbpedia-owl:birthDate ;
  rdf:object  "1961-08-04"^^xsd:date ;
  :group <http://ex.com/group/12> ;
  vrank:hasRank [ vrank:rankValue "3213.101"^^xsd:float ] ] ,

[ rdf:type rdf:Statement ;
  rdf:subject dbpedia:Honolulu ;
  rdf:predicate dbpedia-owl:areaCode ;
  rdf:object "808"@en ;
  vrank:hasRank [ vrank:rankValue "2323.433"^^xsd:float ] ;
  :path [ rdf:type rdf:Statement ;
    rdf:subject  dbpedia:Barack_Obama ;
    rdf:predicate  dbpedia-owl:birthPlace ;
    rdf:object dbpedia:Honolulu ] ] .

<http://ex.com/summary?
entity=dbpedia:Barack_Obama&topK=2&language=en&maxHops=2&
fixedProperty=dbpedia:birthDate,dbpedia:birthPlace#id>
  owl:sameAs dbpedia:Barack_Obama .

dbpedia:Barack_Obama rdfs:label "Barack Obama"@en .
dbpedia-owl:birthDate rdfs:label "birth date"@en .
dbpedia:Honolulu rdfs:label "Honolulu"@en .
dbpedia-owl:areaCode rdfs:label "area code"@en .
dbpedia-owl:birthPlace rdfs:label "birth place"@en .
<http://ex.com/group/12> rdfs:label "Important Dates"@en .
```

References

1. Assaf, A., Atemezing, G.A., Troncy, R., Cabrio, E.: What are the important properties of an entity? In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC Satellite Events 2014. LNCS, vol. 8798, pp. 190–194. Springer, Heidelberg (2014)
2. Berners-Lee, T.: Read-Write Linked Data, August 2009. http://www.w3.org/DesignIssues/ReadWriteLinkedData.html (accessed November 26, 2012)
3. Pietriga, E., Bizer, C., Karger, D.R., Lee, R.: Fresnel: a browser-independent presentation vocabulary for RDF. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 158–171. Springer, Heidelberg (2006)
4. Harth, A.: VisiNav: A system for visual search and navigation on web data. Web Semantics: Science, Services and Agents on the World Wide Web **8**(4) (2010)
5. Käfer, T., Harth, A.: Billion Triples Challenge data set (2014). http://km.aifb.kit.edu/projects/btc-2014/
6. Krummenacher, R., Norton, B., Marte, A.: Towards linked open services and processes. In: Berre, A.J., Gómez-Pérez, A., Tutschku, K., Fensel, D. (eds.) FIS 2010. LNCS, vol. 6369, pp. 68–77. Springer, Heidelberg (2010)
7. Qian, R.: Understand your world with bing (2013). http://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/
8. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media (2007)
9. Roa-Valverde, A., Thalhammer, A., Toma, I., Sicilia, M.-A.: Towards a formal model for sharing and reusing ranking computations. In: Proc. of the 6th Intl. Workshop on Ranking in Databases In conjunction with VLDB 2012 (2012)
10. Singhal, A.: Introducing the knowledge graph: things, not strings (2012). http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html
11. Speiser, S., Harth, A.: Integrating Linked Data and services with Linked Data Services. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 170–184. Springer, Heidelberg (2011)
12. Stadtmüller, S., Speiser, S., Harth, A., Studer, R.: Data-fu: a language and an interpreter for interaction with read/write linked data. In: Conference on World Wide Web (2013)
13. Sydow, M., Pikua, M., Schenkel, R.: The notion of diversity in graphical entity summarisation on semantic knowledge graphs. Journal of Intelligent Information Systems, 1–41 (2013)
14. Thalhammer, A., Knuth, M., Sack, H.: Evaluating entity summarization using a game-based ground truth. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 350–361. Springer, Heidelberg (2012)
15. Thalhammer, A., Rettinger, A.: Browsing DBpedia entities with summaries. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC Satellite Events 2014. LNCS, vol. 8798, pp. 511–515. Springer, Heidelberg (2014)
16. Thalhammer, A., Toma, I., Roa-Valverde, A.J., Fensel, D.: Leveraging usage data for linked data movie entity summarization. In: Proc. of the 2nd Int. Ws. on Usage Analysis and the Web of Data (USEWOD2012) Co-Located with WWW 2012, Lyon, France, 2012, vol. abs/1204.2718 (2012)

17. Torzec, N.: Yahoo's knowledge graph (2014). http://semtechbizsj2014.semanticweb.com/sessionPop.cfm?confid=82&proposalid=6452
18. Verborgh, R., Steiner, T., Van Deursen, D., Van de Walle, R., Valls, J.G.: Efficient runtime service discovery and consumption with hyperlinked RESTdesc. In: Proceedings of the 7th International Conference on Next Generation Web Services Practices (NWeSP 2011), Salamanca, Spain (2011)
19. Wilde, E.: REST and RDF granularity (2009). http://dret.typepad.com/dretblog/2009/05/rest-and-rdf-granularity.html
20. Danyun, X., Cheng, G., Yuzhong, Q.: Preferences in wikipedia abstracts: Empirical findings and implications for automatic entity summarization. Inf. Process. Manage. **50**(2), 284–296 (2014)