How to create easily a data analytic semantic portal on top of a SPARQL endpoint: introducing the configurable Sampo-UI framework

Heikki Rantala[1], Annastiina Ahola[1], Esko Ikkala[1] and Eero Hyvönen[1,2]

[1]*Semantic Computing Research Group (SeCo), Aalto University, Finland*
`https://seco.cs.aalto.fi, firstname.lastname@aalto.fi`
[2]*Helsinki Centre for Digital Humanities (HELDIG), University of Helsinki, Finland*

Abstract
Sampo-UI is a JavaScript framework for creating semantic portals on top of SPARQL endpoints. This paper presents a new, declarative configuration-based version of Sampo-UI that can be used to build very quickly and easily applications with powerful semantic search and browsing functionalities integrated seamlessly with data-analytic and visualization tools. The framework is openly available on Github and has been used successfully in multiple in-use applications in various domains of Digital Humanities. This paper shows how the framework can be used in practise to build a working customised semantic portal in a few hours, using the DBpedia SPARQL endpoint as an example case.

1. Introduction

Sampo-UI [1] is a JavaScript (JS) framework that can be quickly[1] configured to create web applications with powerful search, browsing, and data-analytic visualization tools based on SPARQL[2] queries to RDF[3] triplestores. The user interface (UI) with its components can be created by re-using and configuring a generic UI template and a set of existing UI components whose usage are specified declaratively in JSON configuration files. Using the framework, a first working application with faceted search and various interactive visualization, such as maps and charts, can be created in only a few hours for an existing RDF service with a SPARQL API even with limited programming skills.

Based on the Sampo model [2], a Sampo-UI interface contains a landing page with a set of "application perspectives" to choose from. Each perspective provides the end-user with a faceted semantic search view to filter out individuals of a class (e.g., Person, Place, Artifact, etc.) related to the perspective. After finding a set on individuals of interest, the user can either study and analyze them 1) one by one, 2) analyze sets of individuals together (e.g., statistically, on maps,

0000-0002-4716-6564 (H. Rantala); 0009-0008-6369-4712 (A. Ahola); 0000-0002-9571-7260 (E. Ikkala); 0000-0003-1695-5840 (E. Hyvönen)

[1]To illustrate this, in the video https://vimeo.com/manage/videos/817028609 a simple, but working portal is created in less than an hour.

[2]https://www.w3.org/TR/sparql-query/

[3]https://www.w3.org/RDF/

or on timelines), or 3) explore the data by browsing.

Our earlier article [1] describes the idea, model, and architecture of the initial version of Sampo-UI in detail. In contrast, this paper introduces a new version of Sampo-UI. The biggest change is the new way of configuring the portals using JSON[4] files. The main contribution of this article is to show how "Sampo" portals can be configured with JSON files, how the SPARQL queries are formed, and how the various data analyses and visualizations can be created using Sampo-UI. To test and evaluate the new Sampo-UI version, it has been used to create some 15 portals in use in the Sampo series[5] [2] of Linked Open Data (LOD) systems that demonstrate practical usability of the framework. The example portal[6] of the framework is mainly based on the Mapping Manuscripts Migrations [3] portal. In this paper we use examples from a simple portal which is created in the tutorial available for Sampo-UI. The portal uses the DBpedia [4] endpoint to search writers.

The paper is organized as follows. First, related works are discussed. After this requirements for the Sampo-UI framework are presented and its UI logic is explained. It is then presented how a semantic Sampo portal can be created using the framework and its configuration files. In the final section, contributions of the paper are summarized and discussed and direction for future work is outlined.

2. Related work

The possibilities and benefits of systems utilizing faceted search have long been talked about, especially in the digital library domain [5, 6]. With the rise of linked data, various systems implementing faceted search capabilities on top of of RDF data have been developed. The Ontogator [7] browser was an early implementation of a system with a semantic faceted search over RDF data. The OntoViews semantic web portal tool [8] made it possible to create semantic portals with faceted search by utilizing the Ontogator as a search engine. The OntoViews tool led to the development and publication in 2004 of the first Sampo portal, MuseumFinland [9].

The logic of faceted search with SPARQL queries is described extensively in [10]. Multiple faceted search systems and their faced ranking methods have been surveyed in [11]. Applying faceted search to DBpedia has been implemented before for example in [12]. SPARQL Faceter [13] is a tool for applying faceted search with SPARQL queries fully in the client side of an application[7]. How SPARQL Faceter has been applied for implementing faceted search and visualizations is presented in [13, 14]. In contrast to SPARQL Faceter and many other faceted search engines, the faceted search capabilities are not the only focus of Sampo-UI: the other focus is on data analytical tools to analyze and visualize the data using a variety options, such as charts, timelines, maps, and network analytics.

There exist multiple faceted search systems that facilitate faceted search of RDF data. For example, SemFacet [15] is a prototype of a system for faceted search of RDF data that can be configured to an extent through a graphic UI. A recent example is Rhizomer [16], which

[4]https://www.json.org/

[5]For a full list of Sampo systems see: https://seco.cs.aalto.fi/applications/sampo/

[6]https://sampo-ui.demo.seco.cs.aalto.fi/

[7]DBpedia writers have been also used to demo the use of SPARQL Faceter. See: https://github.com/SemanticComputing/sparql-faceter-dbpedia-demo.

automatically generates facets for exploration and certain visualisation, such as word clouds, from the data. Rhizomer is already used in certain research projects to explore and understand the research data. In contrast to these kind of systems Sampo-UI is mainly aimed for developers to use as a basis for developing new systems that employ faceted search and visualisations.

There have been also been portals with faceted search functionality in the cultural heritage (CH) domain that integrate data analytical tools for the user. The Nomisma portals[8] in the numismatic domain, for example, provide ways of performing faceted search on numismatic data as well as visualizing it on maps. In the field of archaeology, the ARIADNE portal[9] similarly not only offers the faceted search capabilities but also a few ways of visualizing the data in different formats, such as on a map as well as on a timeline. The ResearchSpace [17] tool of British Museum aims to be a comprehensive platform for conducting CH research. ResearchSpace offers not only tools for exploring and visualising the data, but also tools for updating and adding to the data.

3. A framework for creating data-analytic semantic portals

The Sampo-UI framework is based on experiences gathered is developing the so-called Sampo[10] series of LOD services and portals for collaborative CH LOD publishing and research, based on the "Sampo model" [2]. This development work has been going on for over twenty years spanning numerous CH projects and over twenty Sampo systems[11] either in use online or in development. The main goal of developing Sampo-UI has been to make it easier for both the users and developers of new applications by trying to standardize the process of creating new applications as well as the way their user interfaces work.

Requirements for the framework Based on the Sampo model principles, Sampo-UI was designed and implemented with the following major requirements or goals in mind:

1. Create a *consolidated, "standardized" UI logic for semantic portals*, based on Semantic Web standards and best practices of W3C. In this way the portals would be easy use for end-users and to implement and maintain for the developers.

2. Create *semantically rich functionalities* for semantic search and semantic browsing (data exploration), integrated seamlessly with data-analytic and visualization tools. A particular application area of research in our work has been Digital Humanities. [18]

3. The framework should be *re-usable on top different external SPARQL endpoints* without touching the underlying data or LOD service. In this way the tool could be used, e.g., on top of LOD services on the Web, such DBpedia, Wikidata, Getty thesauri, and the CH LOD services of the Linked Data Finland platform[12] hosting the KGs of different Sampo systems.

[8]http://nomisma.org/about/

[9]https://portal.ariadne-infrastructure.eu/

[10]The name 'Sampo' comes from the Finnish epic Kalevala: it refers to a mythical artefact and is a metaphor of ancient technology that brings wealth and fortune to its owner.

[11]Sampo portals homepage: https://seco.cs.aalto.fi/applications/sampo

[12]https://ldf.fi

4. The framework should be *adaptable* to conform to different metadata data models [19] and ontologies used in knowledge graphs, such as Dublin Core, CIDOC CRM, etc.

5. The framework should be *extendable* with new data-analytic and visualization tools needed in different applications and domains.

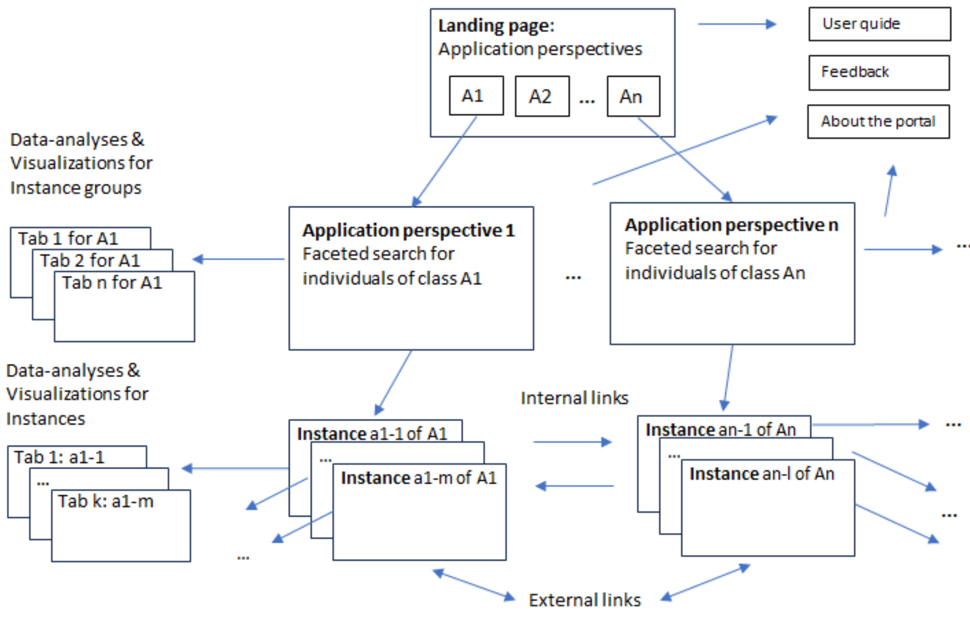


Figure 1: The default navigational page structure of a portal based on Sampo-UI.

Generic UI logic of Sampo-UI Fig. 1 illustrates the navigational structure of using a Sampo-UI-based portal. The user first lands on the *landing page* with several *application perspectives* to the data. The perspectives are based on classes of the underlying KG, such as Artefacts, Persons, Places, etc. The usage cycle of each perspective can be divided into two steps: 1) filter and 2) analyze. The user first filters the data by using the faceted semantic search [20] tools provided by the portal. The results as well as the facet options are updated after each selection of a facet, making it possible for the user to precisely filter the end-result entities by different aspects, e.g., filtering by birth country as shown in Fig. 2. After filtering the data to the wanted subset, the user can analyze the results set, i.e., a set of instances of the class corresponding to the application perspective, with integrated data-analytic tools available as tabs on the application perspective page. An example of a visualisation can be seen in Fig. 2, where the number of publication per year of writers (based on the perspective) in DBpedia KG who have been born in United States (based on the faceted selections). It is easy to see that the number of publications peaks in 2008, and then declines. This could then be compared to selections, such as writers from other countries or from different genres.

It is also possible to select a particular instance of the result set for a closer look: each instance has an instance page that provides aggregated information about the individual with internal and external links for further information to browse. In addition, instance pages also may have a set of tabs that provide contextualized visualisations of the individuals. This filter-analyze two-step usage cycle allows an iterative approach to exploring the data by being able to organically find potentially interesting subsets in the data without having to already be somewhat familiar with it.

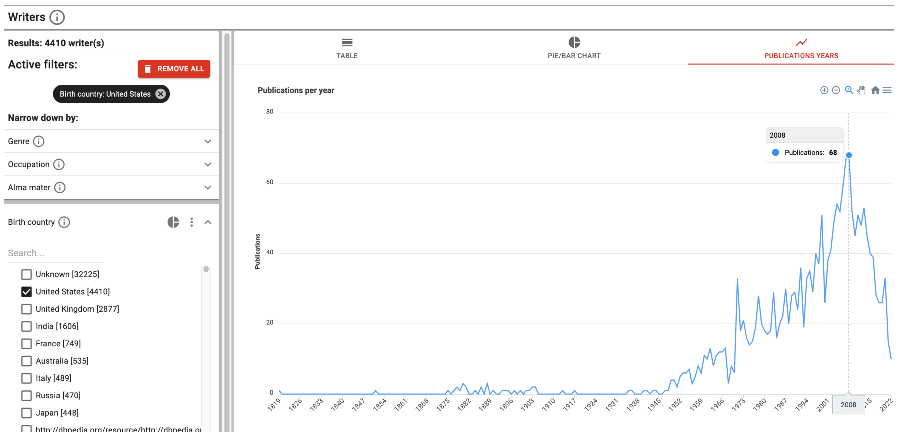


Figure 2: Analyzing writers in DBpedia KG using Sampo-UI: “United States” has been selected on the “Birth country” facet, and the “publication year” line chart option has been selected to visualise the result set as a timeline.

4. Creating a portal using Sampo-UI

This section presents the configuration of a new Sampo-UI portal[13] built to work on top of an existing triplestore database accessible through an open SPARQL endpoint, DBpedia[14]. The demo portal presented in this paper also has a live demo available[15]. The portal is intended as a simple tutorial[16] portal, that can be created quickly to learn how to use Sampo-UI.

The application UI with its components are specified declaratively using JSON files in three main directories: 1) `configs`. JSON files configuring the portal and its perspectives. 2) `sparql`. SPARQL queries referred to in the `configs` files. 3) `translations`. Translations of things like menu items and labels for different locales. When creating a new UI configuration, existing UI components, such as those for the facets and visualization tabs, can be re-used, and the system can also be extended with new components. The Sampo-UI framework consist of a back-end based on Node.js[17] and a JavaScript front–end, based mainly on React and Redux libraries.

Faceted search with SPARQL queries SPARQL is a query language meant to retrieve and change data from RDF databases. Sampo-UI application logic is based largely on creating

[13]Source code for the portal available at: https://github.com/SemanticComputing/dbpedia-sampo-ui-demo
[14]https://www.dbpedia.org/
[15]Available at: https://sampo-ui-dbpedia.demo.seco.cs.aalto.fi/en/
[16]You can see the tutorial at https://seco.cs.aalto.fi/tools/sampo-ui/Sampo-UI-tutorial.pdf. The tutorial portal can be created in around an hour by an experienced user, but a first time learner should reserve more time.
[17]https://nodejs.org/en

SPARQL queries based on facet selections made by a user, and then showing the results in appropriate formats to the user. Generally Sampo-UI only requires SPARQL access to the data through an endpoint. The data only needs to follow general principles of RDF, so that most of the features of Sampo-UI framework can be implemented for it. The main exception to this is free text search which does require specific indexing from the endpoint.

Sampo-UI includes two different implementations for faceted search: "client side faceted search" and "server side faceted search". By client side faceted search we refer to an implementation where some set of data is retrieved from an endpoint through a SPARQL query, transmitted to an internet browser of a user, and searched and visualizations are then performed within the browser to that data without any further need for queries to the SPARQL endpoint. This approach has advantages and disadvantages. This kind of faceted search scales well and will generally be quick, however there is a limit for how much information the client browser can process before running out of memory. This means that faceted search can be applied to only a limited number of instances. Either the data need to be relatively small, or there needs to be some sort initial filtering of data before applying faceted search. In contrast the server side faceted search is implemented through SPARQL queries to an endpoint. This means that there are no limits to the number of instances beyond the limitations of the endpoint. However, the way Sampo-UI implements this kind of search means that every facet and all results require separate SPARQL queries that can be demanding for the endpoint.

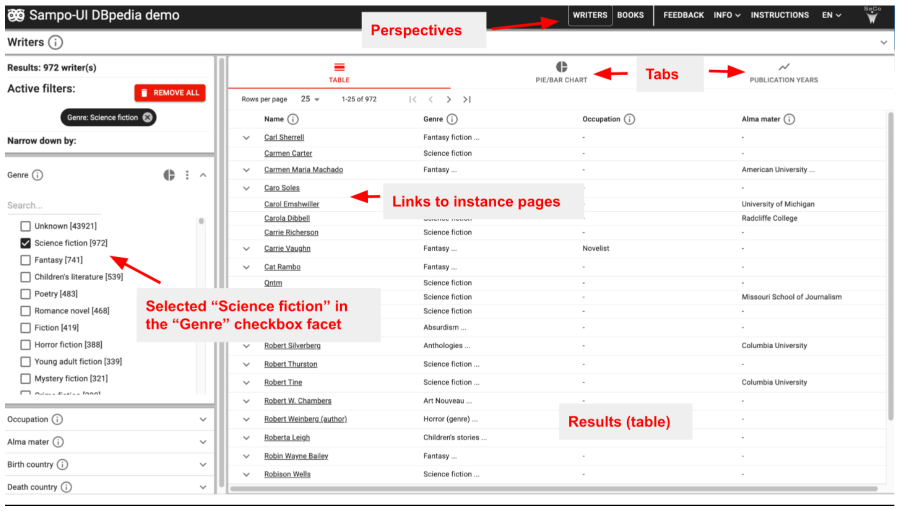
The SPARQL queries are automatically formed mainly using string forms where parts, for example "<FILTER>" or "<ORDER_BY>" are filled by Sampo-UI based on configuration or facet selections using string replace function. The resulting string is combined with prefix file given separately. The results of the query may then be passed through a mapper function before delivering the results to the front-end as a JavaScript object.

Different types of queries used include: *facet values query*, *result count query*, *paginated results query*, *all results query*, and *results by URI query*. In a faceted search perspective generally after every change in facet selections a result count query, a facet values query for every open facet, and some results query is performed. Facet queries get the available selection of values and getting the hit counts for each. Result count query is performed separately, and some kind of results query that can vary depending on what kind of results visualization is selected. Usually as a default the results are shown as a paginated table.

Configuring application perspectives The configuration files for the portal in `src/configs` are divided into two levels: the portal level configuration affecting the whole portal (e.g., list of available perspectives and locales, navigation bar configuration, etc.) and configurations specific to single perspectives (e.g., what columns or facets are included in a specific perspective). The perspective configuration files then reference the name of the correct JS file containing the wanted SPARQL queries as well as the appropriate variable name that holds the specific query to be used in that file.

The JS files with the SPARQL queries contain variables holding various queries to be used for the perspective. The exact format of the queries is dependent on their usage and whether that usage passes them to a template (e.g., facet query template) or not; the queries for getting the properties to be shown as results should just include the inside block of the SPARQL SELECT queries while queries for visualizations should have the whole query. The developer can define all the prefixes they wish to use in the queries and the query structure itself is going to be similar to how one would normally write queries for that particular endpoint. In order for the queries in Sampo-UI to work, the data in the SPARQL endpoint needs to just follow general principles of RDF, with the exception of the free text search functionality, which requires a specific indexing configuration from the endpoint.

The key difference in the queries is how the wanted data is captured in variables. The variable name itself should match the names defined in the configuration files, but the type of the data matters as well. Literal values can be just captured to the variable of same name, but other types, e.g., objects, should be captured as objects. This is done by capturing both the ID and preferred label of the object under the variable name, e.g., `?genre__id` for the ID of a genre and `?genre__prefLabel` for the preferred label. This way the data is handled as an object by the Sampo-UI, which facilitates the developer to, for example, add links to the column values while still showing the preferred labels.



Results: 972 writer(s)

Active filters:

Narrow down by:

Genre

Search...

- [ ] Unknown [43921]
- [x] Science fiction [972]
- [ ] Fantasy [741]
- [ ] Children's literature [539]
- [ ] Poetry [483]
- [ ] Romance novel [468]
- [ ] Fiction [419]
- [ ] Horror fiction [388]
- [ ] Young adult fiction [339]
- [ ] Mystery fiction [321]

Occupation

Alma mater

Birth country

Death country

Rows per page 25 1-25 of 972

Name	Genre	Occupation	Alma mater
Carl Sherrell	Fantasy fiction ...	-	-
Carmen Carter	Science fiction	-	-
Carmen Maria Machado	Fantasy ...	-	American University ...
Caro Soles	[illegible]	-	-
Carol Emshwiller	[illegible]	-	University of Michigan
Carola Dibbell	Science fiction	-	Radcliffe College
Carrie Richerson	Science fiction	-	-
Carrie Vaughn	Fantasy ...	Novelist	-
Cat Rambo	Fantasy ...	-	-
Qntm	Science fiction	-	-
[illegible]	Science fiction	-	Missouri School of Journalism
[illegible]	Science fiction	-	-
[illegible]	Absurdism ...	-	-
Robert Silverberg	Anthologies ...	-	Columbia University
Robert Thurston	Science fiction ...	-	-
Robert Tine	Science fiction ...	-	Columbia University
Robert W. Chambers	Art Nouveau ...	-	-
Robert Weinberg (author)	Horror (genre) ...	[illegible]	-
Roberta Leigh	Children's stories ...	[illegible]	-
Robin Wayne Bailey	Fantasy ...	-	-
Robison Wells	Science fiction ...	-	-

Figure 3: The faceted search view of the demo portal with annotated elements.

Fig. 3 illustrates the structure of the faceted search view in the demo portal using the DBpedia data. The perspective is configured for browsing and searching the writers (`dbo`[18]`:Writer`) present in the DBpedia KG with information on their names, the genre(s) they have written (`dbo:genre`), their occupations (`dbo:occupation`) as well as their almae matres (`dbo:almaMater`). The property path can also be longer as in the case of birth countries (`dbo:birthPlace/dbo:country`).

Each row in the table represents a singular writer with the values in different columns listing that particular information for the specific writer. Clicking on the name of any of the writers would open up that particular writer's own instance page showing the information about that writer as table with each row representing a property and its value.

The genres, occupations, almae matres and birth countries are available as facets to be used for filtering the writers. The facets utilize Sampo-UI framework's checkbox facet type, where each of the facet values for a property is shown in a list with checkboxes next to the label.

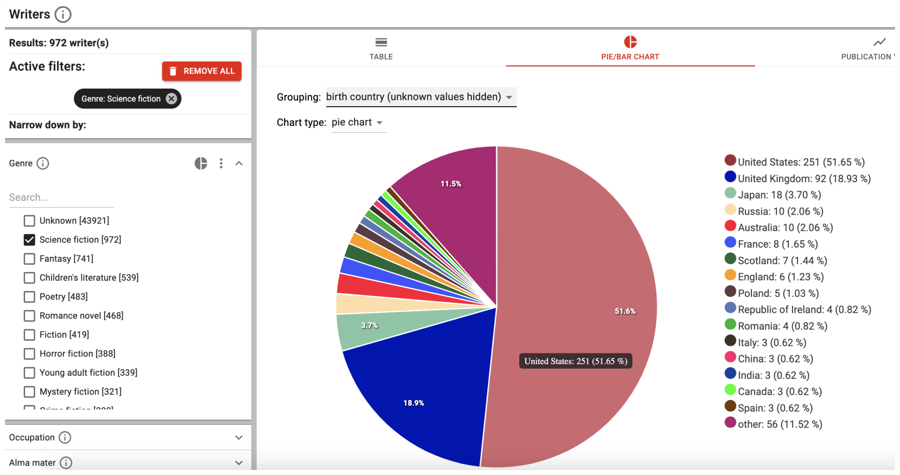
[18]The prefix 'dbo' is used for http://dbpedia.org/ontology/

Selecting a checkbox automatically updates the results and the available facet values in the other facets. Selecting multiple checkboxes inside a single facet updates the results to include all results that match at least one of the facet values, while selecting checkboxes from different facets in a conjunctive manner. For example, selecting the *Science fiction* and *Fantasy* facet values for the genre and the *Novelist* facet value for the occupation would filter the results to include only writers who are novelists by occupation *and* have written *at least one* of the selected genres, science fiction *or* fantasy works.

The values inside the facets can be sorted in two ways: 1) in an alphabetical order or 2) based on how often the value occurs in the data. In addition to just the values present in the data, missing values can be aggregated under the label 'Unknown'. The instance counts are listed after the label for the facet value inside brackets and are updated accordingly if the user makes selections in other facets. The users can use the instance counts in the facet to already get an idea of the nature of the data (e.g., most popular genres or occupations for writers) they are browsing. The inclusion of the 'Unknown' category additionally helps the users to gauge the reliability of the data: High 'Unknown' values indicates low annotation rate for this particular property, which means that the top non-'Unknown' values might now represent the data set as a whole. Sampo-UI framework also includes the option to add a pie and/or bar chart button to the facet to automatically generate a pie/bar chart based on the values of that particular facet for visualizing the distribution of relative hit counts. Below is an example of JSON configuration of the 'genre' facet. You can see, for example, that the 'predicate', meaning the property path from the instance of the perspectives class to the facet value, is given as 'dbo:genre', and the 'pieChartButton' option has the value 'true'.

```
"genre": {
    "containerClass": "ten",
    "facetType": "list",
    "filterType": "uriFilter",
    "facetLabelPredicate": "rdfs:label",
    "facetLabelFilter": "FILTER(LANG(?prefLabel_) = 'en')",
    "predicate": "dbo:genre",
    "searchField": true,
    "sortButton": true,
    "sortBy": "instanceCount",
    "sortByPredicate": "dbo:genre/rdfs:label",
    "sortDirection": "desc",
    "pieChartButton": true
},
```

Configuring visualisations For the results view itself, in addition to being able to see the results as a table, there is another tab for visualizing the distributions in the data as pie or bar charts. For example Fig. 4 shows a pie chart visualisation of birth countries of the writers with science fiction genre selected from facets. It is easy to see that the United States is the birth country in slightly more than half of the cases where the data is available, with UK coming second. This can be easily compared to case where there are no facet selections, and while USA is still number one the proportion is much smaller. On the other hand, it is important to note that in most cases this data is not available in DBpedia, as can be easily seen by looking at the 'Unknown' value in the facets. These pie and bar charts function similarly to the ones that can be included in facets, but offer more freedom to the developer in terms of configuration by having configuration options available as well as using custom queries for generating the results in comparison to the query templates used by the facet menu ones.



Results: 972 writer(s)

Active filters: Genre: Science fiction

Grouping: birth country (unknown values hidden)

Chart type: pie chart

Genre

- ☐ Unknown [43921]
- ☑ Science fiction [972]
- ☐ Fantasy [741]
- ☐ Children's literature [539]
- ☐ Poetry [483]
- ☐ Romance novel [468]
- ☐ Fiction [419]
- ☐ Horror fiction [388]
- ☐ Young adult fiction [339]
- ☐ Mystery fiction [321]

- ● United States: 251 (51.65 %)
- ● United Kingdom: 92 (18.93 %)
- ● Japan: 18 (3.70 %)
- ● Russia: 10 (2.06 %)
- ● Australia: 10 (2.06 %)
- ● France: 8 (1.65 %)
- ● Scotland: 7 (1.44 %)
- ● England: 6 (1.23 %)
- ● Poland: 5 (1.03 %)
- ● Republic of Ireland: 4 (0.82 %)
- ● Romania: 4 (0.82 %)
- ● Italy: 3 (0.62 %)
- ● China: 3 (0.62 %)
- ● India: 3 (0.62 %)
- ● Canada: 3 (0.62 %)
- ● Spain: 3 (0.62 %)
- ● other: 56 (11.52 %)

Figure 4: A pie chart visualisation of the distribution of birth countries of writers of science fiction genre. USA is the birth country in more than half of the cases where the data is available.

The data for the visualisation is formed through three different steps: 1) the SPARQL query for fetching the relevant results, 2) the mapper that maps the results to only include the relevant variables as well as possibly containing some needed preprocessing for the data and 3) the functions processing the data to the correct format for the visualization library as well taking possible configuration options into account. The template for the visualisation in Fig. 4 is the following:

```
SELECT ?category ?prefLabel (COUNT(DISTINCT ?writer) as ?instanceCount)
WHERE {
    <FILTER>
    {
        ?writer a dbo:Writer ;
            dbo:birthPlace/dbo:country ?category .
        ?category rdfs:label ?prefLabel .
        FILTER(LANG(?prefLabel) = 'en')
    }
}
GROUP BY ?category ?prefLabel
ORDER BY DESC(?instanceCount)
```

The `<FILTER>` is used for adding the possible filters made with the facets to the query. Because 'science fiction' is selected from the 'genre' facet, and 'writer' is set as the filter target in the configuration file, Sampo-UI will replace the `<FILTER>` in the query with the following lines:

```
VALUES ?genreFilter { <http://dbpedia.org/resource/Science_fiction>  }
?writer dbo:genre ?genreFilter .
```

The system also adds prefixes before executing the final query[19]. If the user adds more filters using the facets, the visualisation is updated accordingly so that the visualised data is the same as the results listed in the table. The results from this query are then mapped with a pie chart

[19]You can test the query on Yasgui: https://api.triplydb.com/s/aUHk_t9Hm.

mapping function that takes the values for `category`, `prefLabel` and `instanceCount` from the query results. This particular visualization doesn't require any additional processing in the mapping process and the results are passed in this form to the relevant data processing functions and the final visualisation component in the front end. In this case the visualisation is created using the ApexCharts[20] library.

5. Discussion and future work

The Sampo-UI framework is based on years of experiences in creating semantic portals based on RDF data for mainly CH domains, often by people who can be, for example, undergraduate research assistants with basic understanding of programming, but with limited professional skills. These experiences showed that in practice it was usually most convenient to take the code of an existing earlier portal as a starting point, instead of creating a completely new application that would just use some libraries for implementing the search and visualisations. In Sampo-UI this idea is embraced. The framework is essentially a working example application that is meant to be changed to suit the needs of the specific use cases. This means that new applications can be created quickly and with very little knowledge of web development. New visualisations and other improvements can be implemented to the Sampo-UI repository, and then pulled to applications created with the framework. The negative side of this is that the Sampo-UI can be a more complicated tool than many use cases would need, and can include often unnecessary dependencies.

After working with humanities researchers in multiple projects, it is striking how much faceted search and simple visualisation can give new insights to the data even to researchers that are intimately familiar with it. For example, many issues with weird or missing values will be immediately apparent by looking at the facets. However, portals created with Sampo-UI, or similar application, should not be expected to fully replace quantitative analysis. The visualisations are usually limited in some ways, and the exact process of creating the visualisation may not be completely to the researcher when using these kinds of web applications. Their best use is to give insights to the data, by allowing "playing" with the data, and quick testing of hypothesis.

The new JSON configuration file-based version of Sampo-UI was created to make Sampo-UI easier and quicker to use. In this form it is possible to create and customise the portal with very limited understanding of web application development. However, it is still necessary to understand the underlying data and SPARQL queries. Also, customising the application will require some knowledge of programming. This can be challenging for many researchers who might benefit of faceted search analysis of their data. For such uses tools like Rhizomer that do not require any programming skills can be more appropriate.

The approach of Sampo-UI has some limitations. It is created to search instances of classes. For example searching instances of every subclass of a certain class would not currently be easy to implement. Also, faceted search through SPARQL queries can be resource intensive, mostly because of calculating the hit counts in facets. The performance of Sampo-UI queries is related to, at least, the number of instances in the data, and the length of the property paths.

[20]https://apexcharts.com/

Huge numbers of instances or long property paths between instances and facet values can slow the performance considerably. Sampo-UI is mainly tested on current versions of Apache Jena Fuseki [21]. Therefore it is possible that some functionalities might not work with other triple stores without alterations. For example, currently to use other than Lucene text search of Fuseki the basic SPARQL queries need to be changed, and this can't be done through configuration alone. Sampo-UI also does not currently support federated queries in faceted search.

One important reason for using JSON as the basis of configuration is that JSON files are easy to generate with programming languages such as Python or Java. This is intended to make it easy to generate the configurations automatically. We are working on creating tools based on Sampo-UI that can generate portals with RDF configuration or straight from tabular files. We are also working on creating tool based on Sampo-UI that would (semi)automatically analyse parts of the data. In the most simple case it would show the user the missing values in the data, but in a more advanced case it might be able to highlight interesting patterns in the data.

Acknowledgments

The work of writing this paper is partly supported by the Semantic Parliament (ParliamentSampo) project of the Academy of Finland, the EU project InTaVia: In/Tangible European Heritage[22], the Jenny and Antti Wihuri Foundation, and Finnish Cultural Foundation. CSC – IT Center for Science has provided computational resources.

References

[1] E. Ikkala, E. Hyvönen, H. Rantala, M. Koho, Sampo-UI: A full stack JavaScript framework for developing semantic portal user interfaces, Semantic Web – Interoperability, Usability, Applicability 13 (2022) 69–84. doi:10.3233/SW-210428.

[2] E. Hyvönen, Digital humanities on the semantic web: Sampo model and portal series, Semantic Web – Interoperability, Usability, Applicability 14 (2023) 729–744. URL: https://doi.org/10.3233/SW-223034. doi:10.3233/SW-223034.

[3] E. Hyvönen, E. Ikkala, M. Koho, J. Tuominen, T. Burrows, L. Ransom, H. Wijsman, Mapping manuscript migrations on the semantic web: A semantic portal and linked open data service for premodern manuscript research, in: Semantic Web. Proceedings of the The 20th International Semantic Web Conference (ISWC 2021), Springer, 2021.

[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semantic web 6 (2015) 167–195.

[5] G. Alemu, B. Stevens, P. Ross, J. Chandler, Linked data for libraries: benefits of a conceptual shift from library-specific record structures to rdf-based data models, New library world 113 (2012) 549–570.

[6] Y. Sure, R. Studer, Semantic Web technologies for digital libraries, Library Management (2005).

[21] https://jena.apache.org/documentation/fuseki2/

[22] https://intavia.eu/

[7] E. Hyvönen, S. Saarela, K. Viljanen, Ontogator: combining view-and ontology-based search with semantic browsing, information retrieval 16 (2003) 17.

[8] E. Mäkelä, E. Hyvönen, S. Saarela, K. Viljanen, Ontoviews–a tool for creating semantic web portals, in: The Semantic Web–ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings 3, Springer, 2004, pp. 797–811.

[9] E. Hyvönen, E. Mäkelä, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, S. Kettula, Museumfinland—finnish museums on the semantic web, Journal of Web Semantics 3 (2005) 224–241.

[10] M. Arenas, B. C. Grau, E. Kharlamov, Š. Marciuška, D. Zheleznyakov, Faceted search over rdf-based knowledge graphs, Journal of Web Semantics 37 (2016) 55–74.

[11] E. Ali, A. Caputo, G. J. F. Jones, A comprehensive survey of facet ranking approaches used in faceted search systems, Information 14 (2023). URL: https://www.mdpi.com/2078-2489/14/7/387. doi:10.3390/info14070387.

[12] R. Hahn, C. Bizer, C. Sahnwaldt, C. Herta, S. Robinson, M. Bürgle, H. Düwiger, U. Scheel, Faceted wikipedia search, in: Business Information Systems: 13th International Conference, BIS 2010, Berlin, Germany, May 3-5, 2010. Proceedings 13, Springer, 2010, pp. 1–11.

[13] M. Koho, E. Heino, E. Hyvönen, Sparql faceter - client-side faceted search based on sparql, in: Joint Proceedings of the 4th International Workshop on Linked Media and the 3rd Developers Hackshop, CEUR Workshop Proceedings, 2016. URL: http://www.ceur-ws.org/Vol-1615, vol 1615.

[14] P. Leskinen, G. Miyakita, M. Koho, E. Hyvönen, Combining faceted search with data-analytic visualizations on top of a sparql endpoint, in: Proceedings of VOILA 2018, Monterey, California. CEUR Workshop Proceedings, Vol. 2187, 2018.

[15] M. Arenas, B. Cuenca Grau, E. Kharlamov, S. Marciuska, D. Zheleznyakov, E. Jimenez-Ruiz, Semfacet: Semantic faceted search over yago, in: Proceedings of the 23rd International Conference on World Wide Web, WWW '14 Companion, Association for Computing Machinery, New York, NY, USA, 2014, p. 123–126. URL: https://doi.org/10.1145/2567948.2577011. doi:10.1145/2567948.2577011.

[16] R. García, J.-M. López-Gil, R. Gil, Rhizomer: Interactive semantic knowledge graphs exploration, SoftwareX 20 (2022) 101235. URL: https://www.sciencedirect.com/science/article/pii/S2352711022001534. doi:https://doi.org/10.1016/j.softx.2022.101235.

[17] D. Oldman, D. Tanase, Reshaping the knowledge graph by connecting researchers, data and practices in researchspace, in: D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, E. Simperl (Eds.), The Semantic Web – ISWC 2018, Springer International Publishing, Cham, 2018, pp. 325–340.

[18] E. Hyvönen, Using the Semantic Web in Digital Humanities: Shift from data publishing to data-analysis and serendipitous knowledge discovery, Semantic Web – Interoperability, Usability, Applicability 11 (2020) 187–193.

[19] M. Zeng, J. Qin, Metadata, Third Edition, ALA Neal-Schuman, Chicago, 2022.

[20] D. Tunkelang, Faceted search, volume 5, Morgan & Claypool Publishers, 2009.