Move Cultural Heritage Knowledge Graphs in Everyone's Pocket

Maria Angela Pellegrino [a,*], Vittorio Scarano [a] and Carmine Spagnuolo [a]
[a] *Dipartimento di Informatica, Università degli Studi di Salerno, Italy*
E-mails: mapellegrino@unisa.it, vitsca@unisa.it, cspagnuolo@unisa.it

Abstract. In the last years, we have witnessed a shift from the potential utility in digitization to a crucial need to enjoy activities virtually as an alternative to in-person experiences. The Cultural Heritage domain heavily invested in digitization campaigns, mainly modeling data as Knowledge Graphs by becoming one of the most successful application domains of the Semantic Web technologies. Despite the vast investment in Cultural Heritage Knowledge Graphs, the syntactic complexity of RDF query languages, e.g., SPARQL, negatively affects and threatens data exploitation, risking leaving this enormous potential untapped. Thus, we aim to support the cultural heritage community (and everyone interested in cultural heritage) in querying knowledge graphs without requiring technical skills in Semantic Web technologies.

We desire to propose an engaging exploitation tool accessible to all without losing sight of developers' technological challenges. This article, first, analyzes the effort invested in publishing cultural heritage knowledge graphs to quantify data on which developers can rely in designing and implementing data exploitation tools in this domain. Moreover, we point out data aspects and challenges that developers may face in exploiting them in automatic approaches. Second, it presents a domain-agnostic knowledge graph exploitation approach based on virtual assistants as they naturally enable question-answering features where users formulate questions in natural language directly by their smartphones. Then, we discuss the design and implementation of this approach within an automatic community-shared software framework (a.k.a. generator) of virtual assistant extensions and its evaluation on a standard benchmark of question-answering systems. Finally, according to the taxonomy of the cultural heritage field introduced by UNESCO, we present a use case for each category to show the applicability of the proposed approach in the Cultural Heritage domain. In overviewing our analysis and the proposed approach, we point out challenges that a developer may face in designing virtual assistant extensions to query knowledge graphs, and we show the effect of these challenges in practice.

Keywords: Community-shared software framework, Question-answering, Virtual assistant, Knowledge graph, SPARQL

1. Introduction

In the last decade, public institutions and private organizations have invested in massive digitization campaigns to create vast digital collections, repositories, and portals that allow online and direct access to billions of resources [1]. Digitization causes an extraordinary acceleration in digital transformation processes [2] that affected any field, from education to business models [3], from health care [4] to cultural heritage (CH) [2]. Focusing on the CH field, public and private organizations have invested in digitizing any form of data to ensure its long-term preservation and support the knowledge economy [1]. According to UNESCO, the term CH includes *tangible* CH, which can be further specialized in i) movable (such as paintings, sculptures, coins, manuscripts); ii) immovable (such as monuments, archaeological sites), and iii) underwater (shipwrecks, underwater ruins), and *intangible* CH, such as oral traditions, folklore, performing arts, rituals. Moreover, it also encompasses *natural heritage*, i.e., culturally significant landscapes, biodiversity, and geo-diversity; and *heritage in the event of armed conflicts* [5].

Nowadays, CH has become one of the most successful application domains of the Semantic Web technologies [6]. Both public institutions (e.g., galleries, libraries, archives, and museums, a.k.a. GLAM insti-

*Corresponding author. E-mail: mapellegrino@unisa.it.

tutions) and private providers modelled and published CH as Knowledge Graphs (KGs), i.e., a combination of ontologies to model the domain of interest and data published in the linked open data (LOD) format [7], both as independent datasets or by enriching aggregators (such as Europeana [8]) [6].

CH as LOD improves data reusability and allows easier integration with other data sources [6]. It behaves as a promising approach in facing CH challenges, such as syntactically and semantically heterogeneity, multilingualism, semantic richness, and interlinking nature [9]. The availability of CH data in digital machine-processable form has enabled a new research paradigm called Digital Humanities [6] and aims to facilitate researchers, practitioners, and generic users to consume cultural objects [9].

However, KG exploitation is mainly affected by i) required technical skills in generic query languages (such as SPARQL) and in understanding the semantics of the supported operators [10] which are too challenging for lay users [10–14] and ii) conceptualization issues to understand how data are modelled [10, 11].

Natural Language (NL) interfaces can mitigate these issues, enabling more intuitive data access and unlock the potentialities of KGs to the majority of end-users [15]. NL interfaces may provide lay users with question answering (QA) functionalities where users can adopt their terminology and receive a concise answer. Researchers argue that multi-modal communication with virtual characters using NL is a promising direction in accessing KGs [16]. Consequently, virtual assistants (VAs) have witnessed an extraordinary and increasing interest as they naturally behave as QA systems. Many companies and researchers have combined (CH) KG and VAs [1, 17, 18], but no one has provided end-users with a generic methodology to generate extensions to querying KGs automatically.

To fill this gap, our *goal* is the definition of a general-purpose approach that makes KGs accessible to all by requiring minimum-no technical knowledge in Semantic Web technologies. VAs usually give the possibility to extend their capabilities by programming new features, also referred to as VA extensions. It implies that (potentially) everyone can implement custom extensions and personalized VA behaviour. However, playing the VA extension creator's role requires programming skills to design and implement the application logic. Moreover, users must be aware that VA extensions are provider-dependent. Therefore, an extension implemented for Alexa will not be directly reusable for other providers, such as Google Assistant.

We desire to empower lay-users by letting them leave VA users' passive position and play the role of VA extensions creator by requiring little/no-technical skills. Thus, we can reformulate the goal of this work as i) enabling QA over KGs (KGQA) by VA and ii) allowing (lay) users to automatically creating ready-to-use VA extensions to query KGs by popular VAs, e.g., Amazon Alexa and Google Assistant. Thus, we propose a community-shared software framework (a.k.a. generator) that enables lay users to create custom extensions for performing KGQA for any cloud provider, unlocking the potentialities of the Semantic Web technologies by bringing KGs in the "*pocket*" of everyone, accessible from smartphones or smart speakers.

To determine the quantity of CH data modeled as KGs on which developers can rely in designing data exploitation tools in this domain, we overview the CH community effort to create, publish, and maintain KGs belonging to any category determined by the CH taxonomy defined by UNESCO. During the performed analysis, we point out which KG aspects and challenges developers may face in designing an automatic approach to exploit CH KGs. This analysis behaves as a starting point to design the proposed domain-agnostic approach to query (CH) KGs via VAs. To materialize this approach, we implement it in an automatic generator of VA extensions provided with KGQA functionalities. We summarize the configuration of the generator and the process to create VA extension in Fig. 1. The generator architecture in Fig. 1

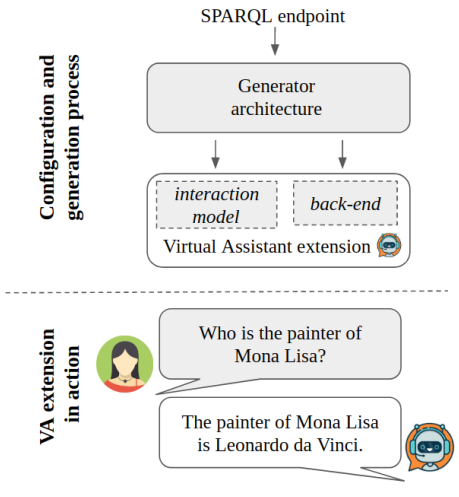

Fig. 1. Overview of the process to configure the generator and create the Virtual Assistant extensions (detailed in Fig. 6) and the extension in action (interaction which will be discussed in Fig. 5).

represents the community shared software framework that will be detailed in Fig. 6. The process starts with a SPARQL endpoint provided by users. The returned VA extension is ready-to-be-use, and it can be used to perform QA, as simulated in Fig. 1 which will be detailed in Fig. 5 to understand the VA extension behaviour fully. To assess the quality of the produced skills and draw out differences in generator configuration options, we design VA extensions for well-known general-purpose KGs, i.e., DBpedia and Wikidata, and we evaluate them on a standard evaluation benchmark for KGQA systems, i.e., QALD. Finally, we overview VA extensions in the CH field as use cases. In particular, we present a skill for each CH data category according to the UNESCO taxonomy to demonstrate the generator in action and show that the proposed approach is general enough to work with any CH data.

The major *contributions* of this paper follows.

- A design methodology to enable lay-users without technical skills in programming and query languages to play the VA extension creator role (Section 4).
- An approach to make KGs compliant with VAs for the KGQA task (Section 4).
- A software tool architecture to automatically generate personalized, configurable, and ready-to-use VA extensions where ready-to-use means that they can be uploaded on VA service providers as manually generated ones (Section 5).
- The open-source release of the software framework *v*1.0 that supports Amazon Alexa, publicly available on the project GitHub repository[1].
- A detailed review and analysis of the CH community effort in publishing KGs and registering them in standard dataset repositories (Section 3).
- The open-source release of a pool of Alexa skills resulting from the generator exploitation to query CH KGs (Section 7 and GitHub repository[1]). We present a use case for each category determined by the UNESCO taxonomy. In particular, for the tangible category, we propose the Mapping Manuscript Migrations (MMM) use case for the movable sub-category, and the Hungarian museum use case for the immovable one; DBTune for the intangible category; NaturalFeatures for the natural heritage category; WarSampo represents the heritage in the event of armed conflict category. We noticed a particular interest in taking care of CH terminology and modeling approaches by thesaurus and model by the performed analysis. Therefore, we also present the UNESCO thesaurus use case representing the Terminology category.

The rest of this article is structured as follows: in Section 2, we overview related work in (CH) KG QA by traditional approaches and by VAs; in Section 3, we quantify the CH community effort in publishing KGs by analyzing the status of the provided services and the amount of published data. This analysis aims to justify the advantages of investing in designing and developing technological solutions to engage lay-users interested in CH and exploit the vast amount of available data in this domain. In Section 4, we detail the proposed domain-agnostic approach to query KGs by VAs by pointing out technological challenges in interfacing KGs and VAs, providing design principles and the implementation methodology, and discussing its strengths and limitations. In Section 5, we overview the VA extension generator that embeds the proposed general-purpose approach in querying KGs, and we assess the performance of the generated skills by evaluating its accuracy on general-purpose use cases (DBpedia and Wikidata) by using standard evaluation benchmarks (the QALD dataset) (Section 6). In Section 7, we present a pool of VA extensions to query CH KGs by showing the general approach in a domain-specific application and by focusing on the impact of the design challenges in the CH context; we conclude with some final remarks and future directions.

2. Related work

QA systems can be classified as domain-specific (a.k.a. closed domain) or domain-independent (a.k.a. open domain). While in domain-independent QA, there is no restriction on the question domain and systems are usually based on a combination of Information Retrieval and Natural Language Processing techniques [19]; in domain-specific QA, questions are bound to a specific context [20] and developers can rely on techniques that are tailored to the domain of interest [21]. Besides the scope, they can be classified by the type of questions it can accept (e.g., facts or dialogs) and queried sources (structured vs. unstructured data) [22]. While systems querying text collections are classified as tools working on unstructured data (e.g., WEBCOOP [23]), systems querying KGs are classified as tools working on structured data. According to this classification, we propose an approach to pose factoids questions (wh-queries, e.g., who, what, and

[1] `mapellegrino/virtual_assistant_generator`. Permanent URI: https://zenodo.org/record/4605951

how many, and affirmation/negation questions) over semantically structured data where questions aim to be as general as possible to classify our proposal as a domain-independent approach.

KGQA is a widely explored research field [24–26]. While it is rare to observe keyword-based questions, most of them address full NL questions. Usually, questions can be posed in English, while some tools deal with European and non-European languages [24]. There is a consistent effort in proposing domain-independent QA systems to query DBpedia and Wikidata [24, 25] by exploiting heterogeneous solutions ranging from combinatorial approaches [24] to neural networks [25], from graph-based solutions [26] to NL request mapping to SPARQL queries [27].

By focusing on **CH KGQA**, i.e., domain-specific systems in the CH domain, they can benefit from many standard data sources. CIDOC Conceptual Reference Model (CRM) is an example in this direction, and it is widely adopted as a base interchange format by GLAM institutions all over the world [28]. CIDOC-CRM has been identified as the knowledge reference model for PIUCULTURA project, funded by the Italian Ministry for Economic Development, which aims to devise a multi-paradigm platform that facilitates the fruition of Italian CH sites. Within the PIUCULTURA project, Cuteri et al. [21] proposed a QA system tailored to the CH domain to query both general (e.g., online data collections) and specific (e.g., museums databases) CIDOC-compliant knowledge sources by exploiting logic-based transformation. As an alternative approach, PowerAqua [29] maps input questions to SPARQL templates under the hypothesis that the SPARQL query's overall structure is almost determined by the syntactic structure of the NL question.

KGQA by VAs is natively offered in well-known VAs, such as Google Assistant and Alexa, that provide users with content from generic KGs (Google Search and Microsoft Bing, respectively). Their main limitations are that i) they query proprietary and ii) general-purpose KGs without exploring domain-specific QA, iii) the provided mechanism can not be extended by users and ported on other KGs. Therefore, the Semantic Web community invested in increasing VA capabilities by providing QA over open KGs. Among others, Haase et al. [30] proposed an Alexa skill to query Wikidata by a generic approach, while Krishnan et al. [31] made the NASA System Engineering domain interoperable with VAs.

By considering **CH KGQA via VAs**, CulturalERICA (Cultural hERItage Conversational Agent) [1] is an intelligent conversational agent to assist users in querying Europeana [8] via NL interactions and Google Assistant technology. Authors state that CulturalERICA is database independent and can be configured to serve information from different sources. Besides technological differences (we opt for Alexa while they opt for Google Assistant), while they enable iterative refinement of the queries, at the moment, we only provide one-step iterations. However, they only enable path traversal, while we also support more complex queries, such as sort pattern, numeric filters, class refinement. Anelli et al. [17] developed a VA to enable the exploitation of the Puglia Digital Library by delegating the speech recognition to Google Assistant. Through subsequent interactions, the VA creates and keeps the context of the request. While they enable keyword-based search, we opt for NL questions. Cuomo et al. [18] proposed an answering system and adapted it to implement a VA able to reply to questions about artworks exposed in Castel Nuovo's museum in Naples. Their proposal aims to reply to questions about artworks, their author, and related information posed by visitors during the touristic tour. Even if it represents an interesting work in the direction of CH KGQA via VAs, it is bound to hardware devices within the museum, and it is not a solution that users can exploit everywhere. About the integration of CH KGs and chatbots, we can cite the chatbot proposed by Lombardi et al. [32] able to support users during an archaeological park visit (i.e., in Pompeii) by simulating the interaction between visitors and a real guide to improve the touristic experience by exploiting natural language processing techniques. In the same direction, Pilato et al. [33] propose a community of chatbots (with specialized or generic competencies) developed by combining the Latent Semantic Analysis methodology and the ALICE technology.

These works behave as evidence of the interest in developing KGQA by VA by promoting interesting applications to make CH KGs interoperable with VAs to accomplish the QA task, but they do not empower end-users by providing them with the opportunity to create their VA extensions. Thus, to the best of our knowledge, the proposed community-shared software framework is the first attempt to provide users without technical skills in the Semantic Web technologies to create KGQA systems via VAs. Consequently, it represents the main *novelty* of our proposal.

3. Cultural Heritage Knowledge Graph Analysis

In this section, we aim to analyze the community effort in producing CH KGs. We aim to make evident the potentialities of proposing exploitation tools in this application domain due to the vast amount of available data. First, we overview the used sources to retrieve the analyzed KGs; second, we provide KG details and a quantitative analysis of available data and, finally, we point out considerations to take into account in proposing an exploitation tool for (CH) KGs.

Selection approach It is worth clarifying that we do not aim to provide a complete overview of all published KGs in the CH context, but the described selection process seeks to point out the absence of bias in the selected KGs and, consequently, the impartiality of the considerations reported in the performed analysis.

According to the following procedure, we exploit the LOD cloud [**?**] and a combination of datasets and articles search engines. Our selection process results in more than `60` KGs covering more than `20` countries.
1 - We exploited the search mechanism provided in the LOD cloud [34] to retrieve KGs containing *museum*, *library*, *archive*, *cultur**, *heritage*, *bibliotec**, *natural*, *biodiversity*, *geodiversity*. It is worth noticing that the search engine requires that the dataset title includes English terms, but it does not pose any constraint on the provider country.
2 - We retrieve datasets registered in the datahub [35] with format equals to `api/sparql`. We manually inspect the `710` returned datasets by looking for *museum*, *library*, *archive*, *culture*, *heritage*, *bibliography*, *natural*, *biodiversity*, *geodiversity*, and similar terms in dataset title and description. Datahub also returns the SPARQL endpoint attached to retrieved datasets. When the specified endpoint is not more available, we search the dataset name attached to "SPARQL endpoint" on the Google search engine to determine if any URL migration took place.
3 - We inspect articles indexed by Scopus and matching the article title, abstract, and keyword filter *("cultural heritage" and ("semantic web" OR "linked data" OR "knowledge graph"))* from 2020 to 2018 (i.e., last two years). It results in `150` articles. We manually check them to verify if they present a KG publication and if so, we further check if authors expose API or a SPARQL endpoint.

KG details According to the UNESCO taxonomy of the CH term, we classify CH KGs according to its content by distinguish *tangible* (further classified as *movable* and *immovable*) (see Tab. 1), *intangible* (see Tab. 2), *natural heritage* (see Tab. 3), and *heritage in the event of armed conflict category* (see Tab. 4). Moreover, we notice a interesting amount of KG dedicated to CH terminology. Therefore, we also consider the *terminology* class, refined as *thesaurus* and *model* (see Tab. 5). If a KG contains elements belonging to multiple classes, we repeat it. For each KG, we report the *original name*, the *country of the provider*, the *service* that enables data exploitation (SPARQL endpoint or API), and the *SPARQL endpoint status* (working or unavailable). For each KG, we also generate a *short name* (mainly combining country and some name keywords clarifying KG content) to refer them in the following analysis quickly. Main observations follow.

World-wide investment. We overview country distribution and CH KG categories of the retrieved collection (see Fig. 2). It is interesting to notice that there is a consistent contribution from European countries, probably due to the vast amount of available raw data and the interest posed in Semantic Web technologies. While Australia and United States gave an interesting contribution to tangible goods, Asian countries also invested in natural heritage. By zooming on Europe (Fig. 2), it is evident that almost every country contributes to CH KGs, mainly in tangible CH. Spain, Netherlands, and Germany can be recognized as main contributors, followed by Italy, England, and Finland. France mainly invested in terminology.

Discontinuous effort. We also considered the ratio between working and discontinued SPARQL endpoints (see Fig. 3). In all the categories, there are SPARQL endpoints that are no more available. In some categories (such as tangible and heritage in the event of armed conflict), discontinued SPARQL endpoints reach almost half of the total available endpoints. Since many endpoints do not work anymore, it shows a discontinuous investment in CH KGs or the lack of attention in updating the dataset search engines when a SPARQL endpoint URL migration occurs.

Investment in all the CH KG categories. There is a substantial interest not only in materializing data but also in defining models (mainly tailored to libraries, archives, and museums [36]) and precise terminology by thesaurus ($10/61 = 17\%$). For instance, the CIDOC-CRM is a theoretical model for information integration in the field of CH. It can help researchers and interested people in modeling CH collections and documents. However, no KG category is ignored. Therefore, data exploitation tools should verify the proposed approach effectiveness by querying

Table 1

Overview of KGs related to *tangible CH*. It contains the *sub-category* interpreted as *movable* and *immovable*, a *short name* of KG to make shorter the following references, the complete *name*, the *country* of the provider, the *Service* that enables the LOD exploitation (SPARQL endpoint or API), and SPARQL endpoint *status* (✓means that it works, while empty cells mean it does not; hyphen means not applicable).

Sub-category	Short name	Name	Country	Service	Status
Movable	ARCO	ARCO	IT	SPARQL	✓
	DigitalNZ	DigitalNZ	NZ	API	-
	Bibliopolis	Bibliopolis	USA	SPARQL	
	Europeana	Europeana	NL	SPARQL	✓
	FondazioneZeri	Fondazione Zeri	IT	SPARQL	✓
	MMM	Mapping Manuscript Migrations	FI	SPARQL	✓
	NL_maritime	Dutch Ships and Sailors	NL	SPARQL	✓
	Nomisma	Nomisma	DE	SPARQL	✓
	Yale	Yale center of British Art	GB	SPARQL	✓
Immovable	DPLA	Digital Public Library of America	USA	API	-
	NZ_museum	Auckland Museum	NZ	API	-
	ADL	Alexandria Digital Library Gazetteer	USA	SPARQL	
	Arc.	Architectural Data	IE	SPARQL	
	ARTIUM	Library and Museum of ARTIUM	ES	SPARQL	
	B3Kat	Libraries of Bavaria, Berlin and Brandenburg	DE	SPARQL	✓
	GB_museum	British museum	GB	SPARQL	
	Cervantes_lib	Biblioteca Virtual Miguel de Cervantes	ES	SPARQL	✓
	CL_library	Biblioteca del Congreso de Chile	CL	SPARQL	✓
	DE_library	Mannheim University Library	DE	SPARQL	
	ES_cultura	Spanish National Library	ES	SPARQL	✓
	ES_library	National Library of Spain	ES	SPARQL	✓
	FI_library	Finnish Public Libraries	FI	SPARQL	✓
	FI_museum	Finish museum	FI	SPARQL	✓
	FR_library	French National Library	FR	SPARQL	✓
	GB_library	British National Bibliography	GB	SPARQL	✓
	GR_library	National Library of Greece Authority Records	GR	SPARQL	
	GR_Veroia_lib	Public Library of Veroia	GR	SPARQL	
	HEBIS	HEBIS – service for libraries	DE	SPARQL	
	Hedatuz	Basque culture and science digital library	ES	SPARQL	
	HU_archive	National Digital Data Archive of Hungary	HU	SPARQL	✓
	HU_museum	Museum of Fine Arts Budapest	HU	SPARQL	✓
	IT_museum	Italian museums	IT	SPARQL	
	JP_library	Japan's National Library	JP	SPARQL	✓
	KR_library	National Library of Korea	KR	SPARQL	✓
	LIBRIS	LIBRIS: Swedish National Bibliography	SE	SPARQL	
	NL_library	Dutch National Bibliography	NL	SPARQL	✓
	NL_archeology	Linked Data Cultural Heritage Agency of the Netherlands	NL	SPARQL	
	Rijksmuseum	Rijksmuseum	NL	SPARQL	
	RU_museum	Russian Museum	RU	SPARQL	
	USA_museum	Smithsonian Art Museum	USA	SPARQL	

Table 2

Overview of KGs related to *intangible CH*. It contains a *short name* of KG to make shorter the following references, the complete *name*, the *country* of the provider, the *Service* that enables the LOD exploitation (SPARQL endpoint or API), and SPARQL endpoint *status* (✓means that it works, while empty cells mean it does not; hyphen means not applicable).

Short name	Name	Country	Service	Status
DBTune	DBTune Western Classical Music	GB	SPARQL	✓
EventMedia	EventMedia	FR	SPARQL	✓
FI_folklore	Semantic Kalevala and Folklore	FI	SPARQL	✓
MusicKG	MusicKG	FR	SPARQL	

Table 3

Overview of KGs related to *natural heritage*. It contains a *short name* of KG to make shorter the following references, the complete *name*, the *country* of the provider, the *Service* that enables the LOD exploitation (SPARQL endpoint or API), and SPARQL endpoint *status* (✓means that it works, while empty cells mean it does not; hyphen means not applicable).

Short name	Name	Country	Service	Status
ARCO	ARCO	IT	SPARQL	✓
EcoPortal	EcoPortal	IT	API	-
Ecology	Linked Open Data of Ecology	TW	SPARQL	
CarbonPortal	Carbon Portal	SWE	SPARQL	✓
NaturalFeatures	Natural Features	GB	SPARQL & API	✓
Ozymandias	Ozymandias	AUS	SPARQL	✓

Table 4

Overview of KGs related to *heritage in the event of armed conflict*. It contains a *short name* of KG to make shorter the following references, the complete *name*, the *country* of the provider, the *Service* that enables the LOD exploitation (SPARQL endpoint or API), and SPARQL endpoint *status* (✓means that it works, while empty cells mean it does not; hyphen means not applicable).

Short name	Name	Country	Service	Status
Munnin	First World War (Muninn project)	CA	SPARQL	
WarSampo	WarSampo	FI	SPARQL	✓

Table 5

Overview of KGs related to *terminology*. It contains the *sub-category* interpreted as *thesaurus* and *model*, a *short name* of KG to make shorter the following references, the complete *name*, the *country* of the provider, the *Service* that enables the LOD exploitation (SPARQL endpoint or API), and SPARQL endpoint *status* (✓means that it works, while empty cells mean it does not; hyphen means not applicable).

Sub-category	Short name	Name	Country	Service	Status
Thesaurus	AAT	The Art & Architecture Thesaurus	CA	SPARQL	✓
	ES_thesaurus	Encabezamientos para las Bibliotecas Públicas	ES	SPARQL	✓
	FR_archive	Thesaurus for Local Archives	FR	SPARQL	
	GB_thesaurus	English Heritage Periods List	GB	SPARQL	✓
	Loanword	World Loanword Database	DE	SPARQL	
	Logainm	Placenames Database	IE	SPARQL	✓
	BNCF	Thesaurus National Central Library of Florence	IT	SPARQL	✓
	UNESCO	UNESCO thesaurus	FR	SPARQL	✓
Model	CIDOC-CRM	CIDOC-Conceptual Reference Model	FR	SPARQL	✓
	MONDIS	Monument Damage Ontology	CZ	API	-

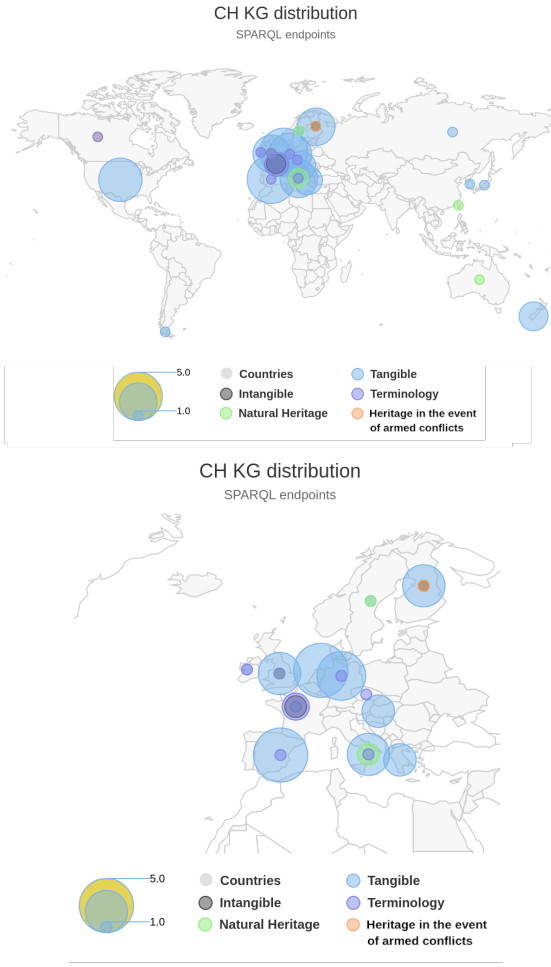

Fig. 2. Geographical distribution of CH KGs. The bubble size represents the number of available CH KGs.

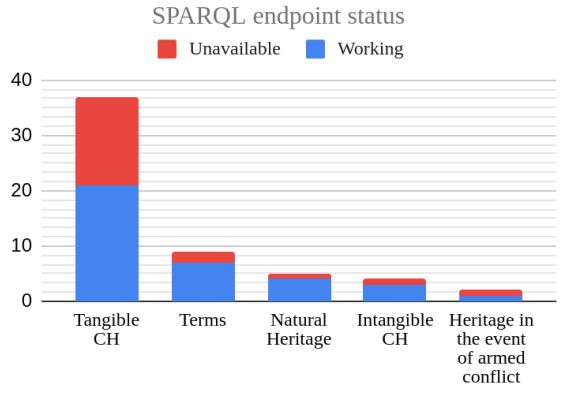

Fig. 3. CH KG SPARQL endpoints status. While blue represents working SPARQL endpoints, red represents unavailable ones.

KG belonging to all the categories to confirm if it is interoperable with any data format and content.

SPARQL endpoints VS API. Few KGs only provide API (8%), while most opt for SPARQL endpoints. Some providers, e.g., Europeana [8], invest in both the access points. Therefore, developers should be aware of available services in designing data exploitation tools to define the best approach to query (CH) KGs. We opt for querying them by SPARQL endpoints as most of the CH KGs configure them.

Quantitative overview of available data. It aims to quantify the data in CH KGs to provide a perception of available sources that can be exploited by automatic data exploitation tools behaving as SPARQL query builders. For each reachable SPARQL endpoint listed in Tab. 1, 2, 3, 4, and 5, we retrieved:

- *classes*, both used classes (the ones returned by the `select count(distinct ?c) where {[] a ?c}` query) and the ones declared as rdfs:Class, skos:Concept and owl:Class. Moreover we also asked for their label (referred to by rdfs:label in all the cases but skos:Concept, where we asked to skos:prefLabel) (see Tab. 6).
- *properties*, both used properties (the ones returned by the `select count(distinct ?p) where {?s ?p ?o}` query) and the ones declared as owl:DatatypeProperty, owl:ObjectProperty, and rdf:property. Moreover we also asked for their label (referred to by rdfs:label in all the cases (see Tab. 7).
- *triples* returned by the `select * where {?s ?p ?o}` query (see column Triples in Tab. 7).

Main observations follow, and they should guide developers in designing automatic data exploitation tools by considering available services' technical constraints.

Label provision. In Tab. 6 and 7, we detail the percentage of classes and properties provided with labels. If developers aim to rely on human-readable labels, they should carefully check them to avoid losing too much data if they only retrieve classes or properties already attached to labels. Some endpoints fail in retrieving labels, such as HU_archieve, KR_library, Nomisma, ARCO, B3Kat, NL_library, NL_maritime, and Yale (check grey lines in Tab. 6 and 7). It evidences a lack of care in attaching human-readable labels to resources by standard approaches, such as rdfs:label. While there is a consistent interest in attaching human-readable labels to classes, properties are rarely pro-

Table 6

Overview of classes in CH KGs (by only considering SPARQL endpoints). It contains the *used classes* and the classes declares as *skos:Concept*, *rdfs:Class* and *owl:Class*. Moreover, it contains the percentage of classes provided with a label (besides its language). Grey lines are endpoints which fail at least a SPARQL query.

Short Name	Used Class		skos:Concept		rdfs:Class		owl:Class	
	TOT	% with label	TOT	% with label	TOT	% with label	TOT	% with label
AAT	75	24	2871894	100	93	25	27	85
ARCO	488	63	30000	100	56	100	615	77
B3Kat	31	0	270	0	18	0	0	0
Cervantes-lib	22	0	64	100	30	0	0	0
CIDOC-CRM	5	40	0	0	102	10	5	80
CL_library	502	39	8334	100	38	100	352	93
DBTune	14	0	0	0	0	0	1	0
ES_cultura	31	52	3	100	15	100	103	99
ES_library	28	14	10000	100	15	100	3	67
ES_thesaurus	2	0	30000	100	0	0	0	0
Europeana	30	13	10000	100	15	100	3	67
EventMedia	50	10	1471	100	56	100	3	67
FI_folklore	17	47	26122	100	0	0	23	87
FI_library	61	0	0	0	0	0	62	0
FI_museum	18	0	0	0	0	0	0	0
FondazioneZeri	105	0	0	0	0	0	0	0
FR_library	38	11	1000000	100	15	100	4	50
GB_library	46	0	1048576	0	0	0	0	0
GB_thesaurus	13	23	500	100	0	0	5	80
HU_archive	469	fail	500	100	384	100	500	100
HU_museum	89	43	10000	100	80	100	129	99
JP_library	5	0	100	0	0	0	0	0
KR_library	53	fail	100	100	0	0	85	33
Logainm	114	4	0	0	57	98	5	40
MMM	58	50	0	0	22	36	124	100
NaturalFeatures	322	90	2519	93	355	100	365	93
NL_library	34	3	113527	100	27	100	0	0
Nl_maritime	92	80	52282	59	131	100	86	90
Nomisma	64	33	107091	fail	4	50	47	70
Ozymandias	30	0	0	0	0	0	0	0
UNESCO	10	20	4427	100	0	0	4	50
WarSampo	90	9	7090	96	84	10	86	3
Yale	43	0	19020	99	52	0	0	0

vided with labels. Developers can complete missing labels by generating them from URI local names. However, this practice can be performed only if KGs adopt human-readable URIs. Lack of label provision is an obstacle to refer and understand resources.

Language support. Multilingualism is a desirable property in the CH community. However, in many cases, labels are defined in just one language (such as in Japanese for JP_library, Spanish for ES_Thesaurus). In some cases, KG providers expose at least labels in the national language and English (such as ARCO CL_library, FI_museum, KR_library). Rare are broader language support; for instance, Nomisma enumerates 177 languages. Moreover, sometimes the language tag is omitted. For instance, GB_thesaurus and Yale are provided with English labels, but if you explicitly ask for `en` as language tag, it returns no results.

SPARQL support. If developers choose to query a SPARQL endpoint or exploit dedicated API directly, they must verify the SPARQL operator support and

Table 7

Overview of properties in CH KGs and triples (by only considering SPARQL endpoints). It contains the *used properties* and the properties declared as *owl:ObjectProperty*, *owl:DatatypeProperty* and *rdf:Property*. Moreover, it contains the percentage of properties provided with a label (besides its language). Grey lines are endpoints which fail at least a SPARQL query.

Short Name	Used Property		owl:ObjectProperty		owl:DatatypeProperty		rdf:Property		Triples
	TOT	% with label	TOT	% with label	TOT	% with label	TOT	% with label	
AAT	43	2	350	100	11	100	490	71	32.094.409
ARCO	945	fail	838	91	244	85	0	0	372.182.177
B3Kat	265	fail	0	0	0	0	0	0	1.022.898.443
Cervantes-lib	128	0	0	0	0	0	0	0	fail
CIDOC-CRM	31	35	0	0	0	0	0	0	5.238
CL_library	357	34	0	0	69	84	0	0	45.413.189
DBTune	38	0	2	0	0	0	0	0	419.519
ES_cultura	354	69	0	0	0	0	0	0	867.535
ES_library	fail	fail	0	0	0	0	0	0	368.989.196
ES_thesaurus	14	0	0	0	0	0	0	0	90.056
Europeana	fail	6	0	0	0	0	0	0	10.000
EventMedia	199	4	0	0	0	0	0	0	11.916.783
FI_folklore	43	19	17	94	7	100	0	0	306.549
FI_library	114	0	0	0	28	0	0	0	4.363.198
FI_museum	60	0	0	0	0	0	0	0	210.986
FondazioneZeri	124	0	0	0	0	0	0	0	fail
FR_library	862	0	0	0	0	0	64	100	fail
GB_library	173	0	0	0	0	0	0	0	204.664.490
GB_thesaurus	51	35	17	100	1	100	28	100	500
HU_archive	fail	fail	2137	23	500	100	0	0	48.378.455
HU_museum	225	19	383	99	20	40	0	0	644.276
JP_library	43	0	0	0	0	0	0	0	21.884.879
KR_library	100	fail	33	100	100	100	0	0	100
Logainm	170	5	0	0	0	0	0	0	1.344.903
MMM	152	24	379	98	9	89	0	0	24.009.834
NaturalFeatures	456	87	116	93	52	96	0	0	918.664.981
NL_library	135	fail	0	0	0	0	0	0	182.580.001
NL_maritime	431	fail	128	100	49	98	590	84	fail
Nomisma	126	23	0	0	61	98	44	0	8.602.910
Ozymandias	196	0	0	0	0	0	0	0	fail
UNESCO	46	0	0	0	3	100	0	0	97.027
WarSampo	310	5	62	5	88	0	0	0	14.322.426
Yale	86	fail	1	0	0	0	93	1	fail

coverage. For instance, AAT, B3Kat, Cervantes_lib, and Ozymandias do not support the `COUNT` operator; JP_library, ES_library, ES_cultura, GB_thesaurus and CIDOC-CRM do not support the `BIND` operator; GB_thesaurus do not support the `DISTINCT` operator. This analysis affects the supported SPARQL patterns in QA applications (e.g., VA extension back-end).

Query failures. Even if some SPARQL endpoints are working, some of them partially or entirely fail in returning results. For example, we fail in retrieving ES_library and Europeana properties. If developers require retrieving the collection of available data, they have to check the way to retrieve them carefully.

Result limit. Some KGs pose a result limit that forces running multiple queries to retrieve all the results. It spans from 100 of KR_library, 500 for HU_archieve to 10000 Europeana. It should be taken into account in verifying the completeness of a single query result.

Running time. We tested SPARQL endpoint execution time by posing 10 times the query to retrieve a

used class (by posing the `SELECT ?c WHERE{[] a ?c} LIMIT 1` query) and the one to retrieve a single triple (by posing the `SELECT * WHERE{?s ?p ?o} LIMIT 1` query). While 24/35 return a class in less than 19s, 3/35 requires a half minute, KR_library requires 2m, ES_thesaurus requires 10m, and 4/35 fails in returning any reply. The triple query execution time returns comparable results to class retrieval run time. The running time may affect the performance of any interactive data exploitation tool. It is crucial to minimize it as much as possible.

4. Question-Answering over Knowledge Graph via Virtual Assistants

This section aims to introduce the design methodology to make KG compliant with VA to address the QA task. We can focus on Amazon Alexa and its terminology without losing generality, as the same considerations can also be adapted for other customizable providers. Alexa VA extensions are named `skills`, and they include both the interaction model and the back-end logic. The interaction model defines the supported features referred to as `intents`, and each intent can be modelled by a set of `utterances`, i.e., phrases to invoke it. Utterances may specify a set of `slot` keywords, i.e., variables that will be instantiated according to the users' requests.

The KGQA task can be defined as follows: given an NL question Q and a KG K, the QA system produces the answer A, which is either a subset of entities in K or the result of a computation performed on this subset, such as counting or assertion replies [37]. We draw a parallel between a general process for KGQA and a VA-based process (see Fig. 4).

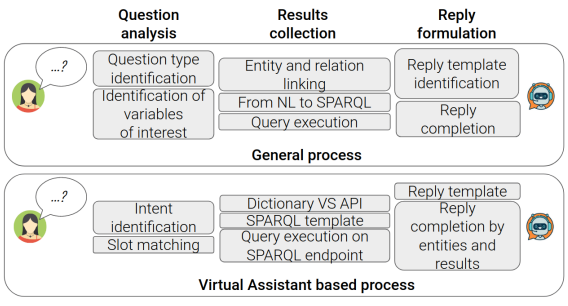

Fig. 4. Parallel of a general and a VA-based KGQA process.

A general KGQA workflow is composed of the question analysis phase, followed by the query construction to retrieve results [38]. We also consider a final step to formulate an NL reply to verbalize the retrieved results and return it to the user. The *question analysis* step performs the question type identification (and consequently, the expected reply template) and the linking phase. The *query construction* phase formulates the SPARQL query corresponding to the NL question and runs it on a SPARQL endpoint to retrieve raw results. During the *reply formulation* step, retrieved results are organized as an NL reply.

In a VA-based process, users pose a question in NL by pronouncing or typing it via a VA app or dedicated device (e.g., Alexa app/device). During the *question analysis* phase, VAs interpret the request and identify the intent that matches the user query by an NL processing component. During the intent identification, VAs also solve intent slots. For instance, suppose that we implement a VA extension representing a thesaurus to recognize questions related to term definition. It might expect requests matching the template `Can you define the term <WORD>?`, where <WORD> is the slot that needs to be completed by the user. Therefore, when the user poses the question `Can you define the term <CULTURAL HERITAGE>?`, where cultural heritage behaves as a slot value. Once retrieved slot values, the VA extension performs the linking step to retrieve the URI(s), which may correspond to the label pronounced by users. The linking phase may be performed by consulting a lookup dictionary or by calling an API service. Completed the question analysis step, we can move to the *query formulation* step. If the KGQA system behaves as a query builder, the VA extension has to recognize the SPARQL pattern that fulfils the user request and formulate the SPARQL query. The SPARQL query can be run on the SPARQL endpoint. Once returned results, the VA extension performs the *reply formulation* step by identifying the reply template corresponding to the activated intent, completing it with actual results, and returning it to the user.

4.1. Design Challenges

Based on the analysis described in Section 3 and the overviewed KG aspects and issues, we identified the following challenges that must be faced in designing VA extensions to enable KGQA.

Label retrieval. According to LD principles [39], every resource must be referred to by a URI. Moreover, KG curators are encouraged to specify human-

readable labels to make these URIs understandable by humans. It is crucial for making them callable by VA-based data exploitation tools. To easily configure systems able to automatically querying KGs, it is required to exploit a uniform (and standard) property to attach human-readable labels to resources. Most of the KGs attach label by standards properties, such as `rdfs:label`, `skos:prefLabel` or `foaf:name`. However, some KGs use domain-specific and custom label-properties (e.g., EventMedia uses `rnews:headline`) that makes the label retrieval step even more challenging.

Label coverage. Developers have to carefully check the percentage of URLs provided with labels (a.k.a. coverage) to avoid losing a high rate of resources by retrieving only URIs attached to human-readable labels.

Label readability. If labels contain codes (e.g., in HU_museum) or are wrongly formatted (e.g., labels are in camel notation, such as hasDate, hasUnit, shipType in NL_maritime), it is hard to recognize the desired resources when pronounced by humans.

Multilingualism. Language support is a desirable property. However, in many cases, labels are defined in just one language. It limits the use and exploitation of available sources.

Label ambiguity. If the same label is attached to several resources, it implies an ambiguous reference to a source of interest. For instance, if Apple is both used for the company and the fruit, it will be up to the VA back-end to solve the pronounced label. While it simplifies the question formulation by the user, it undermines the determinism of the question interpretation. A good trade-off must be detected to maintain the interaction as simple as possible without limiting the desired resources' user control.

Linking approach. To determine the URI corresponding to the pronounced label, developers can rely on i) API provided by the KGs (such as Europeana provides search mechanism), ii) named entity resolution (NER) tools to solve entities (and properties), iii) define a dictionary to maintain a list of URI for each label of interest, or iv) a combination of them. It affects the complexity, reliability, and size of the back-end. While the dictionary guarantees complete control of the entity and property resolution, it requires developer effort and highly affects the back-end size. As pointed out in the analysis described in Section 3, few CH KGs are provided with APIs. Concerning NER, it is a general solution to solve entity label, but i) it rarely works on properties, ii) it is hard to configure NER tools to work on KGs different from the one they are developed for, and iii) it strongly affects the reliability of the VA extension under the definition.

SPARQL support. If developers choose to query a SPARQL endpoint or exploit dedicated API directly, they have to check SPARQL operators' support and coverage in defining the mapping between NL and SPARQL queries in the QA tools.

Running time. Requests execution time strongly affects data exploitation tool performance.

Results limit. Results limit posed by KG services must be carefully checked since a low limit can compromise the completeness of the queries and require several questions provided by the `OFFSET` operator to have a complete reply.

4.2. Principles and Methodology

This section describes the proposed approach to design and implement a VA extension to enable KGQA by focusing on Amazon Alexa as a VA provider. It details the introduced concepts related to Alexa skills and the proposed implementation of a KGQA skill. It is not a loss of generality since it can be easily adapted to any other VA that enables custom skill definition, such as Google Assistant, or in bot implemented by Microsoft Azure Bot Service or Googlebot.

Amazon Alexa skills. As stated before, functionalities in Alexa are named skills. Among the supported types of skills, we are interested in *custom* skills where we can define the requests the skill can handle (intents), and the words users say to invoke those requests (utterances) [40]. By developing new skills, you have to define: a set of intents that represent actions that users can do with your skill; a set of sample utterances that specify the words and phrases users can use to invoke your intents; an invocation name that identifies and wake-ups your skill; a cloud-based service that accepts and fulfills these intents. By mapping utterances to intents, you are defining the skill interaction model. Utterances can contain slots, i.e., variables bound by users when formulating their requests, that can be validated by attaching to each slot a list of valid options during the interaction model definition. The back-end code can be either an AWS Lambda function or a web service. An AWS Lambda (an Amazon Web Services offering) is a service that lets you run code in the cloud

without managing servers. When the user poses a question, Alexa recognizes the activated intent and communicates to your code both the recognized and slot(s) values. Then, the back-end can perform any necessary actions to collect results and elaborate a reply [40].

Virtual Assistants for Question-answering. We model each supported SPARQL query template as an intent. The implemented intents (listed in Table 8) are tailored towards SPARQL constructs, and they mainly cover questions related to a single triple enhanced by the refinement of the subject or object class. More in detail, we cover `SELECT` and `ASK` queries, `class specification`, numeric `filters`, `order by` to get the superlative and path traversal. In table 8, we report, for each intent, an exemplary NL query that activates the intent, the intent name, an utterance by specifying slots among braces, and the related SPARQL triples. In defining utterances, we aim to separate the supported SPARQL patterns clearly to enable users to assess the query correctness generated out of their input. We also avoid utterance overlapping to ensure, as much as possible, a deterministic intent activation.

When the end-user poses a question, Alexa identifies the activated intent and notifies the back-end by communicating both the activated intent and the slot(s) values. For instance, in the CH use case reported in Fig. 5, the user asks for Mona Lisa's painter. The VA recognises that it corresponds to the `getPropertyObject` intent with utterance *what/who is the {property} of {entity}* and attaches to the property slot the value *painter* and to the entity slot the value *Mona Lisa*.

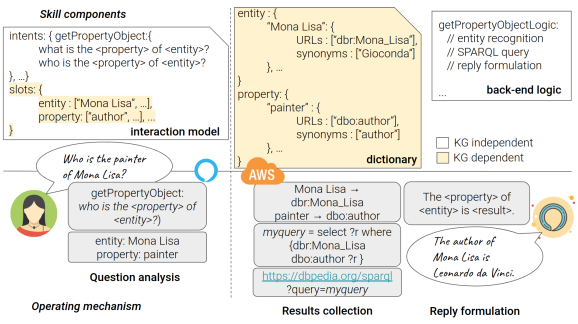

Fig. 5. It is a graphical representation of the skill component where the yellow components are KG dependent and the skill in action in a CH use case by querying DBpedia.

Consequently, the entity and relation linking phase must be performed. It is worth noticing that the performed task is a simplified version of the more general entity and relation linking problem. Entity linking is generally referred to as identifying in a text snippet entities and matching these to the corresponding KG entity. For instance, mapping in the question *Who is the wife of the mayor of Rome?* the textual evidence of *Rome* has to be isolated first, and then it can be mapped to the corresponding KG entity. In our case, named entity textual evidence is already detected by VAs, and we have only to map the named entity textual evidence to a KG node (like *Rome* to the node in the graph representing the city of Rome). To perform this (simplified) linking phase an alternative is performing a dictionary lookup. In such a case, we store the mapping label-URIs in a dictionary by querying KG classes, predicates, and resources URIs and the corresponding labels. The skill back-end exploits the dictionary to retrieve the URI(s) corresponding to NL labels. Resolved entities and predicates are used to complete the SPARQL template. We attach to each intent a different SPARQL query template. Consequently, any NL query posed by end-users is matched to the corresponding intent (according to the VA interaction model), and each intent corresponds to a SPARQL query template (according to our approach). To reconstruct the complete SPARQL query corresponding to each intent, you can proceed as follows: you have to introduce the SPARQL triple(s) reported in Table 8 with the SELECT operator and append the optional request of the label attached to the variable of interest. For instance, the triple `<e><p>?` corresponds to the SPARQL query *SELECT DISTINCT ? ?label WHERE{* `SPARQL triple` *} OPTIONAL { ? <label>?label. FILTER(LANG(?label)="en")}* (supposing that the skill language is English). The notation `<e>` means that the triple is completed by URIs attached to the label `e` in the dictionary. Once the query has been formulated, it can be posed to the SPARQL endpoint. We opt for running a GET query on the SPARQL endpoint and by asking for results in the JSON format. Once results are returned, the back-end formulates them as an NL reply. We attach to each intent a reply template. The back-end completes it with the resolved entities and with the retrieved results. The complete reply, i.e., the reply that includes the resolved entities, enables the end-users to inspect how the system interpreted the performed question implicitly. For instance, in the CH use case in Fig. 5, the end-user acknowledges that the *painter* word has been interpreted as *author*. It behaves as a step forward in the direction of the explicability of the application back-end logic.

Table 8

List of implemented intents by detailing an example that activates the intent, the intent name, an exemplary utterance where slots are represented among braces, and the SPARQL triple used in the SPARQL query formulation step.

Intent name	Utterance	SPARQL Triple
	What is the {author} of {Mona Lisa}?	
getPropertyObject	What is the {p} of {e}?	<e><p>?
	What is {cultural heritage}? Can you define {cultural heritage}?	
getDescription	What/Who is {e}?	<e><definition>?
	Where is {Rome}? Where is the {Mona Lisa}?	
getLocation	Where is {e}?	<e><location>?
	Show me {Paris}. Show me {Mona Lisa}.	
getImg	Show me {e}	<e><img>?
	What has {Beethoven} as {author}?	
getPropertySubject	What has {e} as {p}?	? <p><e>
	How many {paintings} are there?	
getClassInstances	How many {e} are there?	? <instanceof><e>
	Which {pianist} were {influenced} by {Beethoven}?	
getPropertySubjectByClass	Which {c} were {p} by {e}?	? <instanceof><c>. ? <p><e>.
	What has been {modifies} {in} {2020}?	
getNumericFilter	What has {p} {symbol} {val}?	? <p>?o. `FILTER`(?o <symbol><val>)
	Which {source} has been {modified} {in} {2020}?	
getNumeriFilterByClass	Which {c} has {p} {symbol} {val}?	? <instanceof><c>. ? <p>?o. `FILTER`(?o <symbol><val>)
	Which is the {creation} with the {maximum} {number of collaborators}?	
getSuperlative	What is the {c} with {sup} {p}?	? <p>?o. `ORDER BY` (?o). `LIMIT` 1
	Can you verify if {intangible cultural heritage} as {folklore} as {narrower}?	
getTripleVerification	Can you verify if {s} has {o} as {p}?	`ASK` <s><p><o>
	Give me all the results	
getAllResultsPreviousQuery	Give me all the results	-

4.3. Discussion of Strengths and Limitations

The proposed approach queries KGs in *real-time* by exploiting up-to-date data and it is entirely *KG-independent*. Fig. 5 makes evident components that must be reconfigured based on the KG of interest and which components can be left unchanged. It is also a *general-purpose* approach and it can be easily adapted to domain-specific applications (see Section 7). As a general process, utterances make no assumption on question interpretation and the application context. The proposed approach is general enough to be exploited both in querying a single KG than in querying multiple KGs by aggregating query results in the reply formulation step, which means improving the back-end implementation without modifying the general approach. We aim to further investigate multiple KGs in the future.

5. Automatic Virtual Assistant Extensions Generator

This section aims to overview the architecture and implementation of the proposed software framework to automatically generate VA extensions implementing KGQA by requiring little/no-technical skills in programming and query languages.

Our framework provides users with the opportunity to customize VA extension capabilities and generate

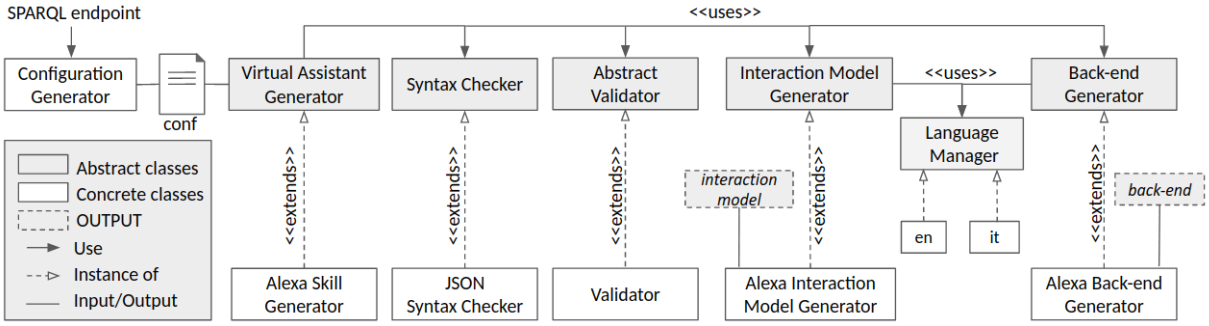

Fig. 6. Architecture of the proposed generator of Question Answering over Knowledge Graphs by Virtual Assistants.

ready-to-use VA extensions. Each phase is kept separate by satisfying the modularity requirement, and it is implemented as an abstract module. The proposed generator architecture is reported in Fig. 6.

VA generator input: the configuration file. The `VA Generator` module takes as input a configuration file containing the VA extension customization options: the `invocation name`, i.e., the skill wake-up word; the list of desired `intents`, according to supported intents listed in Table 8; the `SPARQL endpoint` the user aims to query; the `lang`, by choosing among *en* and *it* at the moment, even though further languages can be easily introduced. Moreover, users can specify a (incomplete) dictionary of entities and properties mapping URIs to labels.

Users can manually create the configuration file. Otherwise, they can exploit the `Configuration Generator` module that takes as input the SPARQL endpoint of interest and automatically retrieves both classes and properties labels and URIs. It looks for used classes/properties and the ones defined according to standard approaches, such as classes defined as `owl:Class` or `rdfs:Classes`, properties defined as `rdf:Property` or `owl:DatatypeProperty`. Moreover, it also expands labels with synonyms and variations by exploiting Wordnet, e.g., nouns used as properties are expanded by their verbal or adjective forms. The configuration file is returned as output, and it can be directly used to start the VA extension generation process. Users can manually check the auto-generated configuration file before generating the VA extension to revise supported resources.

Workflow & Output. Once provided the VA Generator module with the configuration file, it can start the generation workflow, i.e., i) it checks the syntactical correctness of the configuration file by the `Syntax checker`; ii) validates the semantic correctness of the configuration by the `Validator`; iii) creates the `interaction_model.json` by the `Interaction Model Generator` containing configured intents, its utterances and the slot values according to the configuration file; iv) generates the back-end code by the `Back-end generator` and it produces the `back-end` (as a ZIP file) containing the back-end logic implementation. While the syntax checker and the validator strictly depend on the configuration file, the interaction model and the back-end generator depend on the VA provider API. As we require a JSON configuration file, the `JSON Syntax Checker` has to verify that the file is a valid JSON file, while the `Validator` checks if all the mandatory fields are provided, and the configuration is consistent. If any error occurs, the generator immediately stops and returns a message reporting the occurred error. If the configuration is properly provided, the generator returns a folder entitled as the skill wake-up word containing the `interaction model` as JSON file and the `back-end` Node.js code as a ZIP file. It is worth noticing that the generated skill is ready to be used, i.e., it can automatically be uploaded on Amazon developer[2] and Amazon AWS[2], respectively. The generated code corresponds to manually created skills but may reduce required technical competencies and development time.

Extension points. In the version presented in this article (*v*1.0), we support the Amazon Alexa provider. Thus, once validated the configuration file, the Alexa skills components (the JSON interaction model and the zip file implementing the skill back-end that can be upload on Amazon AWS) can be created. Thanks to the architecture modularity, it is easy to develop new

[2]Links for Alexa skill deployment: developer.amazon.com and aws.amazon.com

VA providers' support by focusing on the `Back-end generator` implementation.

Concerning the linking phase, it is performed in a dedicated function (as reported in the documentation) to enable end-users (with skills in programming and KG querying) to customize it, e.g., it by calling APIs (as we point out in Section 7). By default, the back-end exploits the (partial) dictionary to perform the linking step. If the slot value is resolved as a list of URIs by the dictionary lookup, it will exploit them during the SPARQL query formulation. Otherwise, the user value is used as-is in the SPARQL query formulation by comparing it with resource labels.

Moreover, developers may add new supported languages by translating utterances in the target language and extend the reply formulation mechanism to return replies in the desired language. At the moment, English and Italian are supported.

To add a new pattern, developers have to model the new intent as a set of utterances (by solving any arising conflict) and extend the back-end logic to formulate the related SPARQL query and the reply.

6. Evaluation

This section aims to assess the quality of the generated VA extensions and test to what extent configuration options affect the returned VA extensions. All the presented skills and the discussed results are online available on the project GitHub repository (see [1]).

6.1. Evaluation Design

Methodology. The following questions (Qs) guided our evaluation process:
Q1 - Are the results achieved by the auto-configured VA extensions comparable with other KGQA systems?
Q2 - To what extent the manual configuration refinement affect results?
Q3 - Which linking approach between the dictionary lookup and API-based approach achieve the best results?

Dataset & Baselines. We relied on a standard benchmark on KGQA systems, QALD, as it contains benchmarks for multiple well-established KGs (i.e., DBpedia and Wikidata), and it tests both simple and complex questions. We consider the QA system joining the challenge as baselines. We refer to official results published in the QALD report. While for DBpedia, we can rely on QALD-9, for Wikidata, we have to consider QALD-7. As systems joining the QALD-7 challenge relied on a different version of Wikidata, we report results achieved by the Wikidata skill generated by the proposed software framework and the updated version of the QALD-7 dataset to enable further comparisons.

Settings. We generated the DBpedia and Wikidata skills by the proposed software framework. The generated skills are different in configuration options (manual VS auto) and linking approach (dictionary VS APIs). Details concerning the generated skills follow.

Manual Configured DBpedia skill. The manual configuration option requires end-users to perform standard queries on the SPARQL endpoint of interest to retrieve all the classes, properties, and resources and to organize them in the JSON format, as described in Section 5. As Alexa requires the specification of custom slot values in the interaction model and poses a constraint on the interaction model size (1.5MB), developers have to query a sub-graph of the KG of interest. In the sub-graph retrieval, we focus on heterogeneous macro-areas. In particular, the entities dictionary contains all the declared classes (750) and 28.5K resources, distributed as follows: 5K people; 5K cities countries, and continents, 2K rivers and mountains related to the geography field; 3K films, 2.5K musical works, and 3K books belonging to the entertainment category; 4K museums and monuments and 1.5K artworks belonging to the art field; 2.5K animals and celestial bodies, related to the scientific field. The property dictionary contains all the declared properties (5K). We took the first results returned by the DBpedia SPARQL endpoint without either applying any sorting option or checking the returned results' relevance. Once we retrieved all the resources and properties, we performed basic cleansing operations, such as lowercasing labels and removing codes as labels to avoid readability issues. We automatically generate the skill.

Auto-configured DBpedia skill. Users can opt for the auto-configuration by specifying the endpoint of interest and the `Configuration generator` automatically creates the configuration file as described in Section 5. The configuration file contains DBpedia classes and property, while it lets skill users freely refer to resources, and the queried labels will be compared against KG resource labels during the query formulation step. Users can either accept the generated file or manually cleansing the configuration file before

generating the skill. It behaves as a check-point to further reduce the human effort and enable end-users to control the skill generation process. The configuration file initialized the VA generator.

Dictionary-based WikiSkill. We queried a sub-graph of Wikidata. It results in a dictionary composed of 2K classes and 28.5K resources, obtained following the same topic distribution described for the manual configured DBpedia skill. The property dictionary contains all used properties (6.5K). We lowercase all the labels, and we remove the ones containing unreadable codes. We add synonyms to entities and properties by retrieving the Wikidata `also known as` property. We generated the skill, and we used as-is without applying any further modification.

API-driven WikiSkill. The generator back-end provides the opportunity to modify the linking approach by affecting a dedicated script to customize back-end logic functions. We rely on wikibase-sdk[3], a library to make read queries to a Wikibase instance (e.g., Wikidata). `searchEntities` enables the opportunity to perform entity (and property) linking by resolving labels given as input. We created the `API-driven WikiSkill` by i) modifying the linking method from the dictionary lookup to the invocation of the `searchEntities` function and ii) the SPARQL query execution with `sparqlQuery` function in the `Dictionary-based WikiSkill` back-end.

Procedure. We perform the evaluation by performing the following steps. Given the QALD (QALD-7 for Wikidata and QALD-9 for DBpedia) question set,
1 - for each question, we manually check if the pose question matches one of the supported intents (according to table 8), or it can be transformed into a chain of supported intents. For instance, the question "What is the time zone of Salt Lake City?" in QALD-9 on DBpedia matches the getPropertyObject intent ("What is the p of e?") where <time zone> plays the role of p and <Salt Lake City> plays the role of e. The question "What is the name of the university where Obama's wife studied?" in QALD-9 on DBpedia can be transformed into a chain of supported intents where first users can ask for "Who is the wife of Obama?" (corresponding to the getPropertyObject intent where wife is the predicate and Obama is the entity) and, then, "What is the school of Michelle Obama?" (corresponding to the getPropertyObject intent where school is the predicate and Michelle Obama is the entity). If not, we skip this question. Otherwise, we continue the procedure.
2 - we check the activated intent, and we formulate the query according to one of the supported utterances.
3 - we query the skill by the adapted question;
4 - all the replies returned by our skill (including empty results) are stored in a JSON file.
5 - We exploit the official system used to evaluate the QA systems joining the QALD challenge, GERBIL, to perform the result assessment.

For Wikidata and QALD-7, the previous procedure required updating replies in the testing set to compile the current Wikidata version (July 2020). We use the updated version of the QALD-7 training dataset[4] and we share it online to encourage further comparison.

Metrics. We follow the standard evaluation metrics for the end-to-end KGQA task, i.e., we report precision (P) and recall (R) and F-measure (F1) at a micro and macro level.

6.2. Results

Configuration options, the DBpedia case. We compared results achieved by (manual and auto-configured) DBpedia skills [5] with the results achieved by the systems that joined the QALD-9 challenge [41] (Table 9). Regardless of the considered configuration approach, we achieve the best results in all the metrics, and we obtain results from 2 to 6 times better than the second-best result obtained by the participants in the challenge (Q1). The achieved results are justified by i) the exploitation of structured NL questions and ii) the possibility to tune the skill initialization according to specific needs by a fine-grained control. End-users can add data of interest in the configuration file, for instance, resources required by the testing dataset that the previous coarse-grained entity selection has not included. While the manually configured skill obtains optimal results due to the user's full control, the auto-configured DBpedia skill provides lay-users with a

[3]Wikibase-sdk: https://www.npmjs.com/package/wikibase-sdk

[4]QALD-7 training set updated to July 2020 Wikidata status `qald-7-train-en-wikidata-July2020Version.json`

[5]Manual configured and auto-configured DBpedia skill results, respectively: http://gerbil-qa.aksw.org/gerbil/experiment?id=202012170018 and http://gerbil-qa.aksw.org/gerbil/experiment?id=202012170019

good starting point to be used with or without manual refinement (Q2).

Table 9
Manual VS auto-configured DBpedia skills and systems joined the QALD-9 challenge. Best results are highlighted in bold.

Tool	Micro results			Macro results		
	P	R	F1	P	R	F1
ELON	0.095	0.002	0.003	0.049	0.053	0.050
QASystem	0.039	0.021	0.027	0.097	0.116	0.098
TeBaQA	0.163	0.011	0.020	0.129	0.134	0.130
wdaqua	0.033	0.026	0.029	0.261	0.267	0.250
gAnswer	0.095	0.056	0.070	0.293	0.327	0.298
Auto-c.	**0.991**	0.197	0.328	0.369	0.358	0.354
Manual c.	0.990	**0.284**	**0.441**	**0.683**	**0.677**	**0.678**

Linking approach, the Wikidata case. The QALD-7 training set contains 100 questions, but 4 questions cannot be more answered. We can reply to 76/96 questions, while the remaining 20 questions are not supported patterns. The `Dictionary-based` skill obtains the best results (see Table 10) due to the full control that users have in solving ambiguities and customize priority in the linking phase (Q3).

Table 10
Dictionary based VS API-driven WikiSkills (WSs) on QALD-7. Best results are highlighted in bold.

	Micro-results			Macro-results		
	P	R	F1	P	R	F1
Dict. WS	**0.989**	**0.946**	**0.967**	**0.736**	**0.747**	**0.739**
API WS	0.954	0.262	0.412	0.664	0.677	0.669

7. Use cases

This section aims to provide an overview of the benefits and challenges in querying KGs by VAs by presenting a pool of Alexa skills for CH KGs. We overview the generator configuration options, and we show the skill in action. The skills back end and its interaction model are freely available on GitHub [1].

We propose a use case for each category of the CH taxonomy defined by UNESCO. In particular, for the tangible category, we propose the Nomisma use case for the movable sub-category, and the Hungarian museum use case for the immovable one; DBTune for the intangible category; NaturalFeatures for the natural heritage category; WarSampo represents the heritage in the event of armed conflict category; the UNESCO thesaurus represents the terminology category.

7.1. Tangible Movable category: MMM

Mapping Manuscript Migrations (MMM) [42] is a semantic portal for finding and studying pre-modern manuscripts and their movements, based on linked collections of the Schoenberg Institute for Manuscript Studies, the Bodleian Libraries, and the Institute for Research and History of Texts. In particular, it models physical manuscript objects, the intellectual content of manuscripts, events, places, and people and institutions (referred to as actors) related to manuscripts.

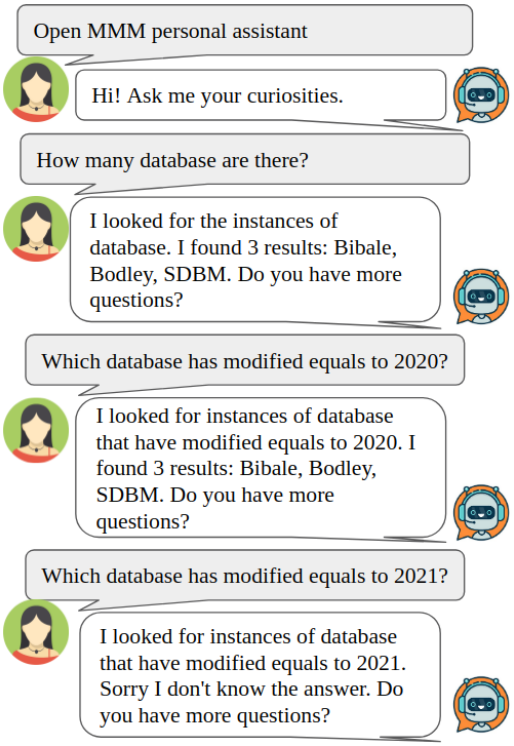

Fig. 7. MMM use case for the tangible category related to the movable sub-category.

Configuration. We automatically configured the MMM skill by exploiting the generator configuration component. The returned configuration file is directly used to initialize the generator.

Skill in action. In fig. 7, you can see a simulation of the interaction between humans and the MMM personal assistant. We ask for databases aggregated by the MMM portal by posing the *How many databases are there?* question. Used resources are i) Bibale (which stands for Bib[liothèque médiév]ale), a long-term project of the Codicological Section of the IRHT (The Institute for Research and History of

Texts) in Paris; ii) Bodley, i.e., Medieval Manuscripts in Oxford Libraries, and iii) SDBM, i.e., Schoenberg Database of Manuscripts. The user request *How many databases are there?* match an utterance attached to the `getClassInstances` intent, which returns the instances of a given class (database in this case). To verify the timeliness of retrieved information, we ask *Which database has modified equals to 2020?* which corresponds to an utterance matching the `getNumericFilterByClass` intent that verifies which instance of a given class (database in our use case) has a property (modified in our case) matching a given numerical value (2020 in our case).

7.2. Tangible Immovable category: Hungarian museum

The Hungarian Museum [43] provides access to the Museum of Fine Arts Budapest data.

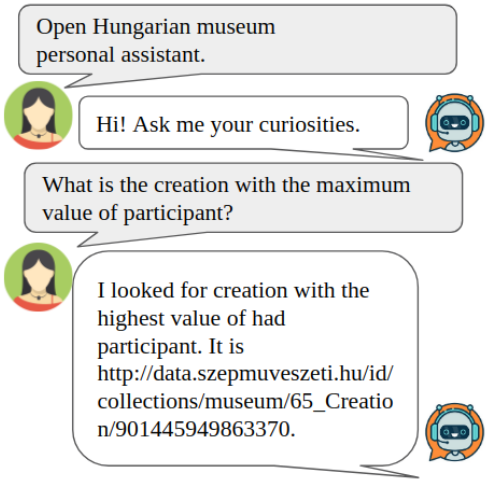

Fig. 8. Hungarian museum use case for the tangible category related to the immovable sub-category.

Configuration. We manually configured the Hungarian museum skill by retrieving `owl:class`, used classes and triples subjects, and the used properties. Labels are mainly provided in Hungarian, without English translation. Moreover, resources often lack any label.
Skill in action. In fig. 8, you can see a simulation of the interaction between humans and the Hungarian museum personal assistant. By querying *what is the creation with the maximum value of had participant* we activated the `getSuperlative` pattern which returns the class instance (the creation in our case) corresponding to the maximum (or minimum) value of a given property (had participant in our use case). The skill usually refers to resources by labels. In this case, it returns the creation URL (see the reply in fig. 8). It makes evident the consequences of lack of labels attached to resources and the difficulties in exploiting them in VA-based applications.

7.3. Intangible category: DBTune classical

DBTune classical [44] describes concepts and individuals related to the Western Classical Music canon. It includes information about composers, compositions, performers, and influence relationships.

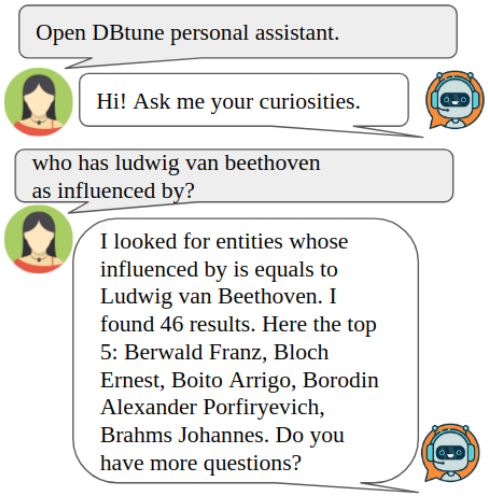

Fig. 9. DBTune classical use case for the intangible category.

Configuration. We automatically configured the DBTune classical skill by exploiting the generator configuration component. The returned configuration file is used to initialize the generator after applying basic configuration manipulation, such as identifying which relation can play the role of label predicate (alias [6] is exploited). This use case demonstrates developers' challenges when the KG adopts a non-standard way to attach human-readable labels to resources.
Skill in action. In fig. 9, you can see a simulation of interaction between humans and the DBTune classical personal assistant. *Who has Beethoven as influenced by?* activates the `getPropertySubject` intent which retrieves the subject of triples where *influenced by* is the property and *Beethoven* is the object.

[6] http://dbtune.org/musicbrainz/resource/vocab/alias

7.4. Natural Heritage category: Natural Features

It is part of Scotland's official statistics [45] that provides a range of statistical and geographic data about Scotland from various organisations. In particular, we are interested in aspects concerning geodiversity, ecology, and biodiversity.

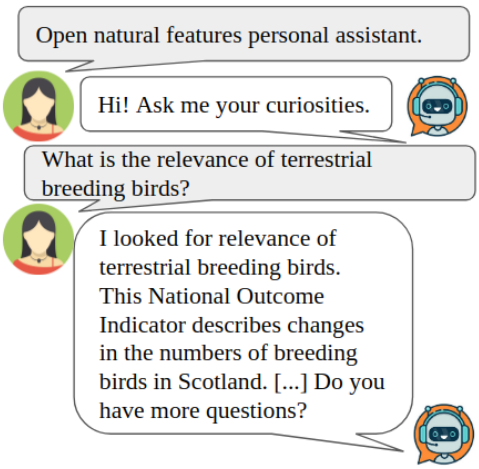

Fig. 10. Natural feature use case for the natural heritage category.

Configuration. We automatically configured the Natural feature skill by exploiting the generator configuration component. The returned configuration file is directly used to initialize the generator.
Skill in action. In fig. 10, you can see a simulation of interaction between humans and the Natural feature personal assistant. *What is the relevance of terrestrial breeding birds?* activates `getPropertyObject` intent which returns the value playing the object role in triples related to *terrestrial breeding birds* as subject and *relevance* as predicate.

7.5. Heritage in the event of armed conflict category: WarSampo

WarSampo [46] includes data concerning the II World War in Finland, including events, actors, places, photographs, and any war aspect. The data covers the Winter War 1939-1940 against the Soviet attack, the Continuation War 1941-1944, where the Winter War's occupied areas were temporarily regained, and the Lapland War 1944-1945, where the Finns pushed the German troops away from Lapland.
Configuration. The entity and predicate dictionary contains all the classes declared as `skos:Concept` and `owl:class`, all the used classes and properties. Labels are mainly provided in Finnish, without providing the English translation.

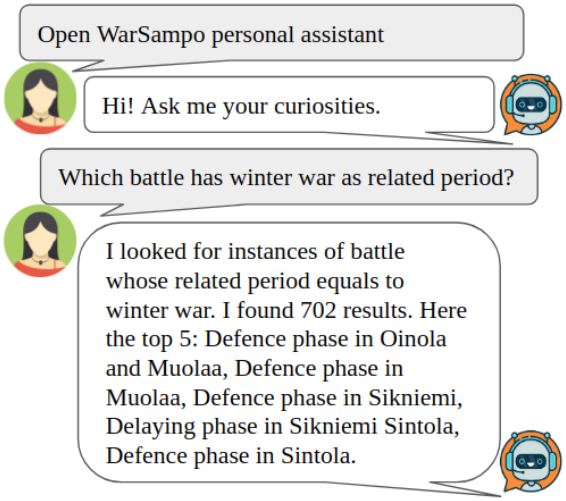

Fig. 11. WarSampo use case for the heritage in the event of armed conflict category.

Skill in action. In fig. 11, you can see a simulation of interaction between humans and the WarSampo personal assistant. WarSampo enables the inspection of temporal correlations among events and the location analysis. To navigate temporal information, we queried *which battle has winter war as related period* by testing the `getPropertySubjectByClass` pattern where *battle* behaves as class, *winter war* as entity, and *related period* as predicate.

7.6. Terminology category: The UNESCO Thesaurus

The UNESCO Thesaurus [47] is a controlled and structured list of terms used in subject analysis and retrieval of documents and publications in education, culture, natural sciences, social and human sciences, communication, and information. Continuously enriched and updated, its multidisciplinary terminology reflects the evolution of UNESCO programs and activities. Like a thesaurus, it mainly provides access to synonyms and related concepts. It also partially behaves like a dictionary by providing term definitions.
Configuration. We manually configured the UNESCO skill by retrieving $(4,421)$ `skos:Concept` that defines all the thesaurus terms and the used properties. All the concepts are attached to a human-readable label

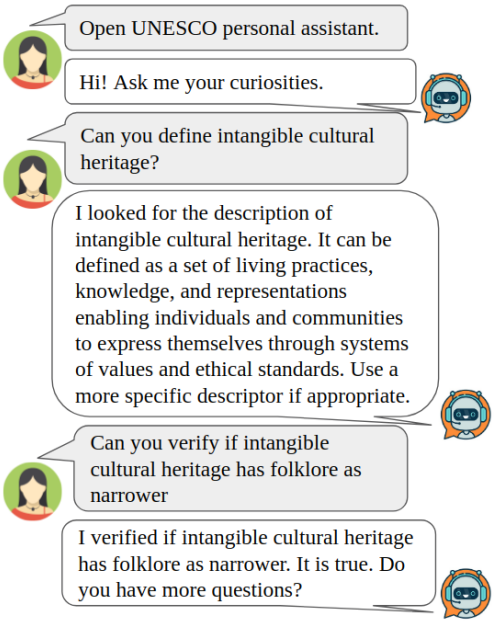

Fig. 12. UNESCO use case for the thesaurus category.

(by `skos:prefLabel`), while we generate property labels by local names of URIs.

Skill in action. In fig. 12, you can see a simulation of the interaction between humans and the UNESCO personal assistant. We can ask for the term definitions, e.g., *what is intangible cultural heritage?* (see Fig. 12). It activates the `getDescription` intent, i.e., a special case of `getPropertyObject` where the property is bound to a relation modelling term description. The skill retrieves the description (configured as skos:scopeNote) attached to intangible CH, and it returns the term definition. We can also pose ask queries. By querying *Can you verify if intangible cultural heritage as folklore as narrower* we activate the `getTripleVerification` pattern, which model ask queries that verify if the stated triple is modelled in the KG.

7.7. Discussion

By the overviewed use cases, we demonstrate most of the intents reported in tab. 8 in real settings. We verify that the proposed approach is general enough to query data concerning different categories of CH, from museums to manuscripts, from music to term definition. Moreover, we also experienced some issues related to some aspects already pointed out in the CH KG analysis (Section 3) and challenges described in Section 4. In the following, we summarize KG properties that affect VA-based KG exploitaiton.

Label coverage. To cope with the scarce provision of human-readable labels, they can be generated by local names of URIs, as we performed in UNESCO Thesaurus. This practice can be performed if resources have human-readable URLs, such as in DBpedia. As evidenced in the Hungarian museum use case, the lack of label provision is an obstacle to resource understanding.

Multilingualism. Some KGs, such as Finland datasets, Hungarian museum, Cultura, only provide access to labels in the data provider native language without enriching resources by English translations. This lack of multilingualism prevents wide exploitation of modeled data.

SPARQL support. A technical detail must be remarked. Before implementing the intents to SPARQL queries mapping, developers have to carefully check if the queried endpoint fully supports SPARQL or omits some patterns. For instance, to use alternative predicates, we exploited the `VALUES` pattern. It is not supported by some of the queried KGs, such as Munnin and CULTURA. It affects the back-end implementation or limits the endpoints that can interface with your implementation. Moreover, there are endpoints, such as CIDOC-CRM and AAT, that do not support the `COUNT` aggregator. It affects queries as simple as *How many artifacts are hosted in the Uffizi museum*.

8. Conclusions

We propose a general-purpose approach to perform KGQA by VA, and we embed it into a community shared software framework to generate VA extensions by requiring minimum/no programming and query language skills. Our proposal may have a significant *impact* as it may unlock the Semantic Web technologies potentialities by bringing KGs in everyone "*pocket*". It is the first attempt in the literature to empower lay-users to create personalized VA extensions, ready to be published on any VA provider.

We propose a *reusable* prototype of VA extensions generator to query any KG. In its actual open-source release ($v1.0$), we allow the building of Alexa extensions, and we aim to provide support for the Google Assistant. It is important to notice that we followed all the best practices in software *design* (e.g., abstraction and modularity) to guarantee *technical quality* and make the generator fully extensible.

The proposed community-shared software framework is *available* on GitHub[1] with an open-source license. The ISISLab research lab of our Department will maintain the code and drive the evolution. We aim to extend the set of supported patterns by formulating iterative queries with consecutive refinements. Moreover, we already plan to evaluate our software framework's usability and user perception in real settings.

References

[1] O.-M. Machidon, A. Tavčar, M. Gams and M. Duguleană, CulturalERICA: A conversational agent improving the exploration of European cultural heritage, *Journal of Cultural Heritage* **41** (2020), pp. 152–165.

[2] D. Agostino, M. Arnaboldi and A. Lampis, Italian state museums during the COVID-19 crisis: from onsite closure to online openness, *Museum Management and Curatorship* **35**(4) (2020), pp. 362–372.

[3] P. Seetharaman, Business models shifts: Impact of Covid-19, *International Journal of Information Management* **54** (2020), pp. 102173.

[4] S. Keesara, A. Jonas and K. Schulman, Covid-19 and Health Care's Digital Revolution, *New England Journal of Medicine* **382**(23) (2020), pp. e82.

[5] The UNESCO Thesaurus, What is meant by cultural heritage?, 2017, Last access March, 2021. http://www.unesco.org/new/en/culture/themes/illicit-trafficking-of-cultural-property/unesco-database-of-national-cultural-heritage-laws/frequently-asked-questions/definition-of-the-cultural-heritage/.

[6] V. de Boer, J. Wielemaker, J. van Gent, M. Hildebrand, A. Isaac, J. van Ossenbruggen and G. Schreiber, Supporting LD Production for Cultural Heritage Institutes: The Amsterdam Museum Case Study, in: *Semantic Web*, 2012, pp. 733–747.

[7] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic Web* **8** (2016), pp. 489–508.

[8] B. Haslhofer and A. Isaac, data. europeana. eu: The europeana linked open data pilot, in: *Int. Conf. on Dublin Core and Metadata Applications*, 2011, pp. 94–104.

[9] E. Hyvönen, Publishing and Using Cultural Heritage Linked Data on the Semantic Web, *Synthesis Lectures on Semantic Web: Theory and Technology* **2** (2012), pp. 1–159.

[10] H. Vargas, C. Buil-Aranda, A. Hogan and C. López, *RDF Explorer: A Visual SPARQL Query Builder*, Springer International Publishing, 2019, pp. 647–663.

[11] P. Bellini, P. Nesi and A. Venturi, Linked open graph: Browsing multiple SPARQL entry points to build your own LOD views, *Journal of Visual Languages & Computing* **25**(6) (2014), pp. 703–716.

[12] D. Damljanovic, M. Agatonovic and H. Cunningham, Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup Through the User Interaction, in: *The Semantic Web: Research and Application*, 2010, pp. 106–120.

[13] S. Ferré, SQUALL: The expressiveness of SPARQL 1.1 made available as a controlled natural language, *Data & Knowledge Engineering* **94** (2014), pp. 163–188.

[14] A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann and D. Gerber, Sorry, i Don't Speak SPARQL: Translating SPARQL Queries into Natural Language, in: *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 977–988.

[15] E. Kaufmann and A. Bernstein, How useful are natural language interfaces to the semantic web for casual end-users?, in: *The Semantic Web*, 2007, pp. 281–294.

[16] P. Cimiano and S. Kopp, Accessing the Web of Data through embodied virtual characters, *Semantic Web* **1** (2010), pp. 83–88.

[17] V.W. Anelli, T.D. Noia, E.D. Sciascio and A. Ragone, Anna: A Virtual Assistant to Interact with Puglia Digital Library, in: *27th Italian Symposium on Advanced Database Systems*, 2019.

[18] S. Cuomo, G. Colecchia, V. Cola and U. Chirico, A virtual assistant in cultural heritage scenarios, *Concurrency and Computation: Practice and Experience* **33**(3) (2021), pp. e5331.

[19] L. Hirschman and R. Gaizauskas, Natural language question answering: the view from here, *natural language engineering* **7**(4) (2001), 275.

[20] A.M.N. Allam and M.H. Haggag, The question answering systems: A survey, *International Journal of Research and Reviews in Information Sciences (IJRRIS)* **2**(3) (2012).

[21] B. Cuteri, K. Reale and F. Ricca, A Logic-Based Question Answering System for Cultural Heritage, in: *Logics in Artificial Intelligence*, 2019, pp. 526–541.

[22] V. Lopez, V. Uren, M. Sabou and E. Motta, Is Question Answering fit for the Semantic Web?: A survey, *Semantic Web* **2** (2011), pp. 125–155.

[23] F. Benamara, Cooperative question answering in restricted domains: the WEBCOOP experiment, in: *Proceedings of the conference on question answering in restricted domains*, 2004, pp. 31–38.

[24] D. Diefenbach, P.H. Migliatti, O. Qawasmeh, V. Lully, K. Singh and P. Maret, QAnswer: A Question Answering Prototype Bridging the Gap between a Considerable Part of the LOD Cloud and End-Users, in: *The World Wide Web Conference*, 2019, pp. 3507–3510.

[25] D. Sorokin and I. Gurevych, End-to-End Representation Learning for Question Answering with Weak Supervision, in: *Semantic Web Challenges*, 2017, pp. 70–83.

[26] L. Zou, R. Huang, H. Wang, J.X. Yu, W. He and D. Zhao, Natural Language Question Answering over RDF: A Graph Data Driven Approach, in: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 313–324.

[27] R.D. Donato, M. Garofalo, D. Malandrino, M.A. Pellegrino, A. Petta and V. Scarano, QueDI: From Knowledge Graph Querying to Data Visualization, in: *Semantic Systems. In the Era of Knowledge Graphs - 16th International Conference on Semantic Systems, SEMANTiCS*, 2020, pp. 70–86.

[28] M. Doerr, The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata, *AI magazine* **24**(3) (2003), 75–75.

[29] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber and P. Cimiano, Template-based question answer-

ing over RDF data, in: *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 639–648.

[30] P. Haase, A. Nikolov, J. Trame, A. Kozlov and D.M. Herzig, Alexa, Ask Wikidata! Voice Interaction with Knowledge Graphs using Amazon Alexa, in: *International Semantic Web Conference*, 2017.

[31] J. Krishnan, P. Coronado and T. Reed, SEVA: A Systems Engineer's Virtual Assistant., in: *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2019.

[32] M. Lombardi, F. Pascale and D. Santaniello, An application for Cultural Heritage using a Chatbot, in: *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, 2019, pp. 1–5.

[33] G. Pilato, G. Vassallo, A. Augello, M. Vasile and S. Gaglio, Expert Chat-Bots for Cultural Heritage, in: *IX Convegno della Associazione Italiana Intelligenza Artificiale Proc. of. Workshop Interazione e Comunicazione Visuale nei Beni Culturali*, 2004, p. 15.

[34] J.P. McCrae, LOD Cloud, 2007, Last access March 2020. https://lod-cloud.net/.

[35] Open Knowledge International, The free data management platform to publish and register datasets, 2006, last access March, 2021. https://datahub.io/.

[36] M. Doerr, *Ontologies for Cultural Heritage*, Springer Berlin Heidelberg, 2009, pp. 463–486, In Handbook on Ontologies.

[37] S. Vakulenko, J.D. Fernandez Garcia, A. Polleres, M. de Rijke and M. Cochez, Message passing for complex question answering over knowledge graphs, in: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1431–1440.

[38] D. Diefenbach, J. Giménez-García, A. Both, K. Singh and P. Maret, QAnswer KG: Designing a Portable Question Answering System over RDF Data, in: *The Semantic Web*, 2020, pp. 429–445.

[39] W3C, LinkedData, 2016, Last access March, 2021. https://www.w3.org/wiki/LinkedData.

[40] A. Developer, Build Skills with the Alexa Skills Kit, 2014, Last access March, 2021. https://developer.amazon.com/en-US/docs/alexa/ask-overviews/build-skills-with-the-alexa-skills-kit.html.

[41] R. Usbeck, R.H. Gusmita, A.N. Ngomo and M. Saleem, 9th Challenge on Question Answering over Linked Data (QALD-9), in: *Proc. of the 9th Question Answering over Linked Data challenge co-located with 17th International Semantic Web Conference (ISWC)*, 2018, pp. 58–64.

[42] Semantic Computing Research Group (SeCo), Mapping Manuscript Migrations, 2017, last access March, 2021. https://mappingmanuscriptmigrations.org/.

[43] Museum of Fine Arts Budapest, Open Linked data from the Museum of Fine Arts Budapest, 2016, last access March, 2021. http://data.szepmuveszeti.hu.

[44] Y. Raimond, DBTune classical, 2007, last access March, 2021. http://dbtune.org/classical/.

[45] Scottish Governmentl, Open access to Scotland's official statistics, 2010, last access March, 2021. https://statistics.gov.scot.

[46] M. Koho, E. Heino, P. Leskinen, E. Ikkala, M. Tamper, K. Apajalahti, J. Tuominen, E. Mäkelä and E. Hyvönen, WarSampo Knowledge Graph on Zenodo, Zenodo, 2019.

[47] UNESCO, The UNESCO thesaurus, 1977, last access May 2020. http://vocabularies.unesco.org/.

[48] W3C - World Wide Web Consortium, SPARQL Query Language for RDF, 2008, Last access March, 2021. https://www.w3.org/TR/rdf-sparql-query/.