TTProfiler: Computing Types and Terms Profiles of Assertional Knowledge Graphs

Lamine Diop, Arnaud Giacometti, Béatrice Markhoff, and Arnaud Soulet

Université de Tours, LIFAT, Blois, France
firstname.lastname@univ-tours.fr

Abstract. As more and more knowledge graphs (KG) are published in the Web, there is a need of tools for abstracting their content for their producers to verify their result, and for their consumers to use it. This implies showing the schema-level patterns instantiated in the graph, with the frequency with which they are instantiated. A profile represents this information. In this paper, we propose a new type of profile that we call TT profile, for Types and Terms profile. It shows the used *Types* and predicates, and also the used *Terms* because of their paramount importance in most of KGs, especially in the Cultural Heritage (CH) domain. We present an algorithm for building a TT profile from an online KG's assertional part, and we report on experiments performed over a set of CH KGs.

1 Introduction

It has become widespread in the Cultural Heritage (CH) field to generate Knowledge Graphs from legacy datasets, using one or more ontologies [2]. A knowledge graph (KG) is a dataset in RDF, i.e. a set of (*subject*, *predicate*, *object*) triples. CH KGs contribute to the Linked Open Data(LOD) construction, publicly offering inter-linked and semantically defined datasets, which is supposed to boost knowledge discovery and efficient data-driven analytics at a world-wide scale. However, using LOD datasets for analysis requires a clear idea of their content and this is a long-standing difficulty. It is not enough to know which ontologies are used, it is necessary to know how they are used, i.e. which of their components serve in that particular dataset, and in what way.

In the last ten years several proposals raised for helping users knowing what contains a given KG, by extracting its predicates, the types of entities they link, and some basic statistics, like ABSTAT [6] does. In the same way, our aim is to generate such an abstract image of the KG *as it is queryable online, by default without reasoning*. From a given KG, ABSTAT builds a set of (C, P, D) triples with statistics, where C and D are types and P is a predicate. Such triple is called Abstract Pattern (AP). Figure 1 (left) shows the four first APs returned by ABSTAT when asking for the predicate `dbo:country` on a 2016 dump of DBpedia in English, using its online tool[1]. The first AP indicates that there are $560,532$

[1] `http://abstat.disco.unimib.it/`

dbo:Location (839987)	dbo:country (770950)	dbo:Country (13169)	560532
schema:Place (1665807)	dbo:country (770950)	dbo:Country (13169)	560532
dbo:Location (839987)	dbo:country (770950)	wgspos:SpatialThing (1965986)	546131
schema:Place (1665807)	dbo:country (770950)	wgspos:SpatialThing (1965986)	546131

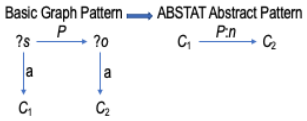

Fig. 1: ABSTAT (left), Basic Graph Pattern and its corresponding AP (right).

RDF triples (last column) in this KG for which the predicate `dbo:country` relates a *subject* of type `dbo:Location` to an *object* of type `dbo:Country`, which informs us that we can query locations and their associated countries. Figure 1 (right) presents the Basic Graph Pattern (BGP) able to compute an ABSTAT AP representing its instances, with n its frequency (number of its instances in the KG). Edges labeled with "a" represent the predicate `rdf:type`.

ABSTAT returns thousands of APs just for the predicate `dbo:country` from this dataset, several of them representing the same facts in the KG. For instance, `dbo:Location` and `schema:Place` in Figure 1 are probably both types of the subjects of predicate `dbo:country` that have objects of type `dbo:Country`, since the two APs have exactly the same frequency (i.e., 560,532). In other words, if the BGP in Figure 2 (a) was instanciated in the KG, then ABSTAT would generate four APs (cartesian product of subject's and object's types), all with the same frequency n. For representing each fact in the KG with only one AP, we propose to deal with APs where predicates relate not just types but *sets of types*, as in Figure 2 (a). Moreover, to the best of our knowledge there is no tool that highlights not only the types of the subject and object of a predicate, but also the *terms* used for objects in the KG. It is one thing to indicate that there are instances of `crm:E22_Man-made_Object` in the graph, but the fact that it contains information about coins, or burials, or garments, is much more interesting and precise. In KGs that use the CIDOC Conceptual Reference Model [2] (hereafter CIDOC), this information is carried by terms, more precisely by URIs described in some thesauri, such as `http://nomisma.org/id/coin` for instance. This is the case because, quoting [3], "CIDOC defines and is restricted to the underlying semantics of database schemata and document *structures* used in cultural heritage and museum documentation in terms of a formal ontology. It does *not* define *any of the terminology* appearing typically as data in the respective data structures; however it foresees the characteristic relationships for its use." The type `crm:E55_Type` is a gateway to these controlled vocabularies, but it is not the only one, for example `crm:E57_Material` can also be detailed in an ad hoc nomenclature. This choice is in line with the use of databases in CH communities insofar as it organises in ontology the entities of the domain and their relationships, but not the descriptive values, i.e. most of the values in databases. In general, these ones are listed and described elsewhere in authority lists, for interoperability

[2] `cidoc-crm.org/`

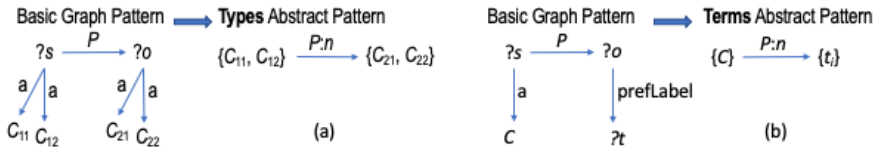

Fig. 2: Types (a) and Terms (b) Abstract Patterns.

purposes. This means that CIDOC-based KGs generally employ various sets of terms, which provide at least as much meaning as the used CIDOC types and predicates. For taking this into account, we propose abstract patterns showing terms used in the graph, as illustrated in Figure 2 (b), where t_i denotes the instances of the variable $?t$ and the edge labeled with "prefLabel" represents the situation where the variable $?o$ is instantiated by a term and $?t$ by a label of that term. Detecting this situation depends on how are implemented the vocabularies. In this paper we consider those implemented with SKOS[3].

To sum-up our contribution, we deal with KGs that are sets of assertional knowledge, whose intentional part is formally defined by existing RDFS or OWL ontologies, and who contain instances of SKOS concepts, defined in existing SKOS thesauri. Given that the ontologies and thesauri used are not necessarily accessible online for programs, we present and discuss a program called `TTProfiler` to build a set of Types and Terms (TT) APs that we call a profile, by querying its online SPARQL endpoint[4]. The rest of this paper is organised as follows: in Section 2, we provide definitions for TT APs and profiles. In Section 3 we present the TTProfiler algorithm. We report in Section 4 our uses of its implementation on various online CH KGs. We conclude in Section 5.

2 Definitions and Problem Formulation

We use Description Logics (DL) [1] formal notations for defining our problem: we consider a Web KG as a knowledge base (KB) $\mathcal{K}$, composed of the TBox $\mathcal{T}$ (names and assertions about concepts and roles, respectively called types and predicates in this paper) and the ABox $\mathcal{A}$ (assertions about individuals, called entities and facts). For instance DBpedia is a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, one example of assertion in $\mathcal{T}$ is `dbo:Artist` $\sqsubseteq$ `dbo:Person`, meaning that the type `dbo:Artist` is subsumed by the type `dbo:Person`, i.e. all artists are persons. $\mathcal{T}$ also includes assertions like $\exists$`dbo:birthYear` $\sqsubseteq$ `dbo:Person`, meaning that the predicate `dbo:birthYear` is defined for persons. On the ABox side, `dbo:Person`(`dbr:Michelle_Obama`) declares that entity `dbr:Michelle_Obama` is a person and `birthYear`(`dbr:Michelle_Obama`, 1964) states the fact that Michelle Obama was born in 1964. Also, some persons are related via the predicate `dct:subject` to a SKOS concept, for instance

[3] `https://www.w3.org/TR/skos-reference/`

[4] `TTProfiler`'s codes are available at `https://github.com/DTTProfiler/DTTProfiler`

we find in DBpedia `Person`(`dbr:Ringo_Madlingozi`), `skos:Concept`(`Category:1964`) and `dct:subject`(`dbr:Ringo_Madlingozi`, `Category:1964`).

Our aim is to give an abstract of the ABox content, which in the Web of data is in general far bigger than the TBox. We work with the asserted KG (or ABox), not with the version one gets by applying a reasoner. By default, SPARQL endpoints do not perform entailments. We put in evidence all the types and predicates appearing in the ABox, whatever the ontologies they belong to, so we do not limit ourselves to only one given ontology. We also want to highlight the SKOS concept instances appearing in the ABox. To do so we look for `skos:Concept` instances or subjects of `skos:prefLabel`, and we also look for declared prefixes that correspond to some known thesauri. We do not use the ontologies and thesauri in the algorithm presented in this paper. When publicly available, they can be used latter on, together with profiles, for completing them. There may be cases in which the TBox is limited to few ontologies that are consistent by themselves and semantically compatible with each other. In those rare cases, a reasoning step combining the TBox and ABox could also be performed before or during the profile generation. This is out of the scope of this paper, because we want our proposal to work on the online Web, which presents too much uncertainty about the quality of the ontologies used (they are not even guaranteed to be accessible online for programs). This is why we limit our scope to the ABox content without considering the TBox, leaving it to other services to process the ontologies and thesauri for enriching the profile information afterwards, when needed. The algorithm presented in this paper builds a *profile* of the given ABox $\mathcal{A}$, that is composed of TT AP (Types and Terms abstract patterns), which are triples whose subjects and objects are sets, as defined in Definition 1. In [6], APs are triples (C, P, D), where C and D are types and P a predicate: we call them *basic APs*. TT APs generalise basic APs in two ways: first, objects can be either types or terms (labels of instances of `skos:Concept`). Second, both subjects and objects are sets (either set of types or set of terms), as illustrated in Figure 2.

Definition 1 (TT Abstract Pattern). *Given an ABox $\mathcal{A}$, a TT abstract pattern of $\mathcal{A}$ is a triple $(\mathcal{C}, P, \mathcal{D})$ such that $\mathcal{C}$ is a set of types in $\mathcal{A}$, P is a predicate in $\mathcal{A}$, and $\mathcal{D}$ is either a set of types in $\mathcal{A}$ or a set of terms appearing in $\mathcal{A}$. Here a term is a string literal, label of an instance of `skos:Concept`. A TT abstract pattern $(\mathcal{C}, P, \mathcal{D})$ represents a fact $P(a, b)$ of $\mathcal{A}$ if:*

- *the entity a is an instance of each type in $\mathcal{C}$ (i.e., $C(a) \in \mathcal{A}$ for $C \in \mathcal{C}$), and*
- *the entity b is an instance of each type in $\mathcal{D}$ (i.e., $D(b) \in \mathcal{A}$ for $D \in \mathcal{D}$) or the entity b is an instance of `skos:Concept` and its prefLabel is in $\mathcal{D}$ (i.e., `skos:Concept`(b) and `skos:prefLabel`(b, t) and $t \in \mathcal{D}$).*

This definition can be declined in various versions. For instance, the subject and object of an AP could be generalised to types not actually appearing in $\mathcal{A}$ but defined using $\mathcal{T}$, as `owl:Thing`, `rdfs:Literal` and so-called *minimal* types used in [6]. Also for instances of `skos:Concept`, one could use some definitions in their respective thesaurus. As already noticed, contrary to [6], if a or b have

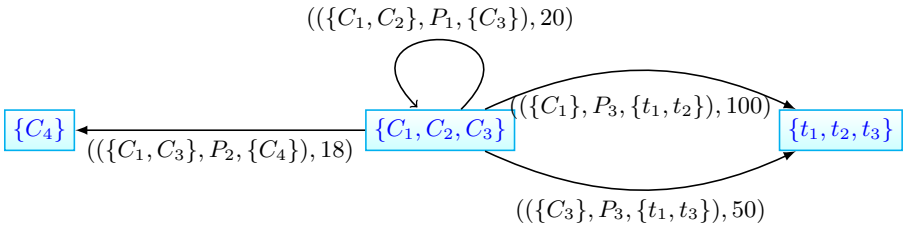

Fig. 3: Graph with maximal sets

several types asserted in $\mathcal{A}$ (whether or not linked in $\mathcal{T}$ by a subsumption) then by Definition 1 the fact P(a,b) is represented by only one AP. Also contrary to [6], a fact $P(a, b)$ having no type asserted for a, or having neither a type asserted for b nor any clue allowing to know that b belongs to a thesaurus, does not raise any AP. Given the set of APs generated from an ABox $\mathcal{A}$ according to Definition 1, we can associate statistics with those patterns, leading to the following definition of a *TT profile*:

Definition 2 (TT Profile). *Given an ABox $\mathcal{A}$, a TT profile $\mathbb{P}$ of $\mathcal{A}$ is a set of pairs $((\mathcal{C}, P, \mathcal{D}), S)$ such that $(\mathcal{C}, P, \mathcal{D})$ is a TT AP generated from $\mathcal{A}$, and S is a statistic value describing $(\mathcal{C}, P, \mathcal{D})$.*

There are many ways to define interesting statistics of a KG's assertional part. We may consider the global number of assertions $C(a)$ for each type C, the global number of assertions $P(a, b)$ for each predicate P, the global number of assertions $P(a, b)$ for each SKOS concept b appearing in $\mathcal{A}$... In this paper, we deal with the frequency of a TT AP, that is how many facts of $\mathcal{A}$ it represents. We call weight the function that associates with $(\mathcal{C}, P, \mathcal{D})$ its frequency in $\mathcal{A}$.

Definition 3 (Weight of a TT abstract pattern). *The weight of the TT abstract pattern $(\mathcal{C}, P, \mathcal{D})$, denoted $\omega((\mathcal{C}, P, \mathcal{D}))$, is the function that associates with $(\mathcal{C}, P, \mathcal{D})$ its frequency in $\mathcal{A}$. $\omega((\mathcal{C}, P, \mathcal{D})) = |\{P(a, b), P(a, b) \in \mathcal{A}$ and $P(a, b)$ is represented by $(\mathcal{C}, P, \mathcal{D})$ according to Definition 1$\}|$.*

Moreover, for the sake of drawing clearly the TT profile as a graph, we aim at grouping the sets of types in such a way that each type appears in only one set (or node). For instance if we have in a TT profile $\mathbb{P}$ the TT APs $\mathcal{A}_1 = (\{C_1, C_2\}, P_1, \{C_3\}))$, $\mathcal{A}_2 = (\{C_1, C_3\}, P_2, \{C_4\}), \mathcal{A}_3 = (\{C_1\}, P_3, \{t_1, t_2\})$ and $\mathcal{A}_4 = (\{C_3\}, P_3, \{t_1, t_3\})$, with $\omega(\mathcal{A}_1) = 20$, $\omega(\mathcal{A}_2) = 18$, $\omega(\mathcal{A}_3) = 100$ and $\omega(\mathcal{A}_4) = 50$, then we merge sets $\{C_1, C_2\}, \{C_3\}, \{C_1, C_3\}$ and $\{C_1\}$ into a maximal set $\{C_1, C_2, C_3\}$ and sets $\{t_1, t_2\}$ and $\{t_1, t_3\}$ into a maximal set $\{t_1, t_2, t_3\}$, which gives the representation shown in Figure 3.

Searching for maximal sets is searching for the components of the graph formed by the profile's nodes (subjects and objects of TT APs), with an edge connecting two nodes if and only if there is a non-empty intersection between

these two nodes. The union of component's nodes is a maximal set. Computing the components of a graph is generally done by a linear depth-first search, but in Algorithm 1 we incrementally compute the *maximal sets* φ during the TT profile building. For the profile visualisation, maximal nodes can be represented by one of their types or terms, and the others can be shown on demand. As shown in Figure 3, edges are annotated with the corresponding AP and its weight.

We can now state our problem as follows: **Given the assertional part of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, how to efficiently generate and visualise a TT profile of $\mathcal{A}$?**

3 `TTProfiler` Algorithm

`TTProfiler` computes a TT profile of an ABox $\mathcal{A}$ following a three steps procedure: 1) basic abstract patterns and statistics recovery, 2) TT profile computing, and 3) TT profile visualisation structure building.

Step 1: Basic abstract patterns and statistics recovery. We recover all basic abstract patterns (C, P, D) with w, their frequency, i.e. the number of instances of (C, P, D) in $\mathcal{A}$ (line 1). An assertion $P(a, b)$ in $\mathcal{A}$ is said to be an instance of the abstract basic pattern (C, P, D) if and only if a is of type C in $\mathcal{A}$ (i.e., $C(a) \in \mathcal{A}$) and b is either of type D or a term of a thesaurus.

Step 2: Profile computing. To fit Definitions 1 and 2, for each predicate appearing in a basic abstract pattern we group all types that have common instances (lines 5-11), and we also group terms for subjects having the same type (lines 12-14). For this last case, we associate to the predicate a weight equals to the sum of the weights computed in Step 1. With the resulting weighted TT APs, each fact P(a,b) is represented by only one pattern. Each computed TT AP is added into the TT profile $\mathbb{P}$ (line 15). We also incrementally compute the set φ of maximal nodes, incorporating in it the nodes $\mathcal{C}$ and $\mathcal{D}$ (that are sets of types or terms) (line 16). The incorporation of a node in φ consists in grouping its elements with other nodes containing them, as explained in Section 2 (cf. the example illustrated in Figure 3).

Step 3: Profile Visualisation structure computing. In this last step, for each weighted TT abstract pattern we replace its subject and object by their corresponding maximal node in φ (lines 19-20) and we add the resulting triple to the Profile Visualisation structure PV.

Regarding the complexity, Step 1 consists in querying the KG, so it depends on the SPARQL endpoint and the network capacities; Step 2 is linear in the number of predicates and quadratic in the number of basic abstract patterns computed in Step 1; Step 3 is linear in the number of TT abstract patterns.

4 Experiments with Cultural Heritage KGs

`TTProfiler`, whose code is published in Github (see Section 1), is devised to apply to KGs that can be queried online via a SPARQL endpoint. This requires

Algorithm 1 `TTProfiler`: Types and Terms Profiler

Input: The ABox $\mathcal{A}$ of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
Output: The TT profile $\mathbb{P}$ of $\mathcal{A}$ and its visualisation structure PV
//**Step 1:** *basic abstract patterns extraction from $\mathcal{A}$ and statistics computation*
Let $R = \{((C, P, D), w)/(\exists P(a, b) \in \mathcal{A}) \wedge C(a) \in \mathcal{A} \wedge (D(b) \in \mathcal{A} \vee (skos:Concept(b) \wedge skos{:}prefLabel(b, D)))\}$ where w is the number of instances $P(a, b)$ in $\mathcal{A}$ for (C, P, D)
//**Step 2:** *TT profile computing: grouping types and terms in sets*
Let $\mathcal{P} = \{P/(\exists((C, P, D), w) \in R)\}$ ▷ $\mathcal{P}$ is the set of predicates in R
$\mathbb{P} \leftarrow \emptyset$, $\varphi \leftarrow \emptyset$ ▷ φ is a set of maximal sets of types or terms
for $(P \in \mathcal{P})$ **do** ▷ grouping types and terms by predicates
  for $((C_1, P, D_1), w_1) \in R$ **do**
    $\mathcal{C} \leftarrow \{C_1\}$, $\mathcal{D} \leftarrow \{D_1\}$, $w \leftarrow w_1$
    for $((C_2, P, D_2), w_2) \in R \wedge ((C_1 \neq C_2) \vee (D_1 \neq D_2))$ **do**
      if $(C_1 \neq C_2) \wedge (D_1 = D_2) \wedge (\forall P(a, b) \in \mathcal{A} : C_1(a) \in \mathcal{A} \wedge C_2(a) \in \mathcal{A})$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_2\}$ ▷ group the types of subjects
      if $(D_1 \neq D_2) \wedge (C_1 = C_2) \wedge (\forall P(a, b) \in \mathcal{A} : (D_1(b) \in \mathcal{A} \wedge D_2(b) \in \mathcal{A})$ **then**
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{D_2\}$ ▷ group the types of objects
      if $(isLabel(D_1)) \wedge (isLabel(D_2)) \wedge (C_1 = C_2)$ **then**
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{D_2\}$ ▷ group the terms
        $w \leftarrow w + w_2$
  $\mathbb{P} \leftarrow \mathbb{P} \cup \{((\mathcal{C}, P, \mathcal{D}), w)\}$
  $\varphi \leftarrow add(\mathcal{C}, \varphi)$, $\varphi \leftarrow add(\mathcal{D}, \varphi)$
//**Step 3:** *Profile visualisation structure*
$PV \leftarrow \emptyset$
for $((\mathcal{C}, P, \mathcal{D}), w) \in \mathbb{P}$ **do**
  $\mathcal{A} \leftarrow maxNode(\mathcal{C}, \varphi)$
  $\mathcal{B} \leftarrow maxNode(\mathcal{D}, \varphi)$
  $PV \leftarrow PV \cup (\mathcal{A}, ((\mathcal{C}, P, \mathcal{D}), w), \mathcal{B})$
return ($\mathbb{P}$, PV)
// $add(\varphi, \mathcal{C})$ returns the set of maximal nodes φ having incorporated $\mathcal{C}$
// $isLabel(D)$ returns true if D is a term, label of a *skos:Concept* in $\mathcal{A}$
// $maxNode(\mathcal{C}, \varphi)$ returns the maximal node that contains $\mathcal{C}$

to carefully write the SPARQL queries in Step 1 because of fair use policies applied by public SPARQL endpoints. Moreover, as already said about the time complexity, Step 1 of computing a TT profile depends on the configurations of the SPARQL endpoint and the network capacities. Considering only the client side computation (Step 2 and Step 3), on small graphs, less than $1,000,000$ triples, the TT profile generation takes about 0.06 seconds. For $91,000,000$ triples it takes 1.15 seconds. `TTProfiler` is implemented in Java using the Jena library to query the public SPARQL endpoints. It was run on Windows 10 with an Intel core i7 processor and 32 GB of RAM.

We design this program as part of a French CH project called SESAMES[5] and we test it with the archaeological KGs grouped in OpenArchaeo[6]. Those graphs are generated from legacy databases, based on a common model which is a small excerpt of the CIDOC and its extensions. Even with such a restricted ontology, all types and predicates do not have instances in all KGs, so the visual query tool that OpenArchaeo provides could be complemented by the display of TT profiles to show what can be asked. In addition, the producers of these graphs use the TT profiles to inspect the results of the KG automatic generation, which is based on mappings expressed with tools like Ontop and X3ML. KG producers exactly know which predicates, types and terms should appear in the TT profiles, and can therefore *easily detect anomalies* in their mappings.

Besides the KGs in OpenArchaeo, we looked for other graphs using the CIDOC CRM and offering a SPARQL API usable by an application. Of those found, many are not always online and many do not answer to counting SPARQL queries of Step 1. We present in Table 1 nine graphs that are currently[7] capable of answering the required queries. Seven of them are from OpenArchaeo (`Kition`, `Iceramm`, `Arsol`, `Epicherchell`, `Outagr`, `Rita`, and `Aerba`) and are rather small, while the `Smithsonian`'s[8] and `Doremus`'s[9] graphs are of different designs, use English terminologies, and are much larger. Doremus contains multilingual labelsTable 1 shows the number of edges and nodes in KGs, the number of distinct types/terms appearing in $\mathbb{P}$, the number of TT APs (i.e., $|\mathbb{P}|$) and maximal nodes (i.e., $|PV|$). Although the set of basic abstract patterns is already a condensed representation of the original graph, it can be too large to be easily visualised, hence the grouping of types and terms and the use of the notion of maximal node, which allows us to display graphs with less nodes, as shown in the last two columns of Table 1. Figure 4 gives an example of the graph visualisation offered to end-users.

Table 2 shows that the nine KGs use the CIDOC and eventually one or more of its extensions (CRMsci, CRMarch, CRMba). Doremus uses the so-called Erlangen implementation of the CIDOC (denoted ecrm). In addition, these KGs use terms of thesaurus, the PACTOLS [10] for OpenArchaeo KGs, and an internal vocabulary for `Doremus`. When thesauri are used, the number of the terms is far larger than that of types. Concerning the predicates instantiated in each KG, here again, all KGs use the CIDOC or its extensions. As can be noticed, CRMba is used by ArSol in OpenArchaeo for one type and not for any predicate: this is because the extensions are attached to CIDOC by subsumption links. So, one can use extension classes with predicates that are defined in CIDOC. In general both types and predicates of extensions are used in the tested KGs.

[5] `http://anr-sesames.map.cnrs.fr/`
[6] `http://openarchaeo.huma-num.fr/explorateur/home`
[7] June, 2021
[8] SPARQL API: `http://edan.si.edu/saam/sparql`
[9] SPARQL human interface: `http://data.doremus.org/sparql`
[10] `https://pactols.frantiq.fr`

Table 1: Knowledge graphs and TT profiles

	statistics for $\mathcal{A}$			statistics for TT profile		
$\mathcal{A}$	nb triples	nb nodes	language	nb types & terms	nb AP	nb nodes
`Aerba`	3,318	1,695	fr	5	3	5
`Epicherchell`	3,488	1,372	-	31	15	13
`Kition`	26,773	9,165	fr	72	31	19
`Iceramm`	32,687	9,325	-	13	21	13
`Rita`	40,479	10,769	-	184	6	7
`Outagr`	79,420	39,573	fr	8	8	8
`Arsol`	670,757	21,2143	-	94	34	17
`Smithsonian`	2,542,142	969,172	en	18	35	18
`Doremus`	91,093,326	24,141,972	en	599	678	146

We present in Figure 4 a visualisation of a small profile, that of `Epicherchell`. In this graph, nodes suffixed by `et_al` are sets of terms and colours denote namespaces (e.g. blue for CRMsci). A click on an edge, here the predicate `P4_has_time-span`, displays its subject and object. Node's content is also displayed on demand, for instance in the bottom of the figure one have selected the node `autel_et_al`. These terms are used to describe the usage of objects. The profile shows that not only the objects are characterised by the type `E57_Material` with the predicate `P45_consists_of`, but it also contains the set of terms that are used in the KG for each material, with the node `albatre_et_al` that appears as object of the same property.

5 Related Works and Conclusion

Compared to works aiming at discovering the schema of a graph, or extracting modules or constraints from KGs, our problem is much simpler. TT profiles can be seen as a special kind of KG summaries as described in the recent and comprehensive surveys on KG summarisation [4]. Its authors classify the existing summarisation techniques in four classes: (i) *structural methods* that consider the paths (quotient graphs) or the subgraphs (with high centrality) in the KG, (ii) *pattern mining methods* that discover patterns in the KG and use them for showing a synthesis of the graph, (iii) *statistical methods*, that extract from the graph quantitative measures or statistics, and (iv) *hybrid methods* that combine some of the previous ones. As explained in [4], the proposals can be distinguished by their inputs and outputs: some works consider only ontologies, some others exploit only instances, and hybrid approaches process both. Outputs also differ ranging from a graph (not necessarily an RDF graph) to a set of items (e.g., rules or queries). Finally, very few of these proposals can be used online or make the source code available. We found only one work [7] that have been applied in the CH field, a structural method focused on centrality of concepts in the ontology. At the time of writing, it is not online usable and we did not succeed to get the sources. The closest to our proposal is ABSTAT [6] dealing with both

Table 2: Types, Terms and Predicates in the TT profiles

	Number of Types						Number of Terms	
$\mathcal{K}$	crm	crmsci	crmarch	crmba	ecrm	doremus	PACTOLS	vocabulary
Aerba	4	0	0	0	0	0	0	0
Epicherchell	9	1	0	0	0	0	21	0
Kition	12	1	1	0	0	0	57	0
Iceramm	11	1	0	0	0	0	1	0
Rita	5	0	0	0	0	0	178	0
Outagr	6	1	0	0	0	0	1	0
Arsol	11	1	1	1	0	0	79	0
Smithsonian	17	0	0	0	0	0	0	0
Doremus	0	0	0	0	15	40	0	459

	Number of Predicates					
$\mathcal{K}$	crm	crmsci	crmarch	crmba	ecrm	doremus
Aerba	3	0	0	0	0	0
Epicherchell	13	2	0	0	0	0
Kition	26	3	1	0	0	0
Iceramm	19	2	0	0	0	0
Rita	5	0	0	0	0	0
Outagr	7	1	0	0	0	0
Arsol	29	3	1	0	0	0
Smithsonian	27	0	0	0	0	0
Doremus	0	0	0	0	112	101

data and schemas, with a set of abstract patterns as output, usable online. To the best of our knowledge, none of the existing summarisation proposals can be tested on SPARQL endpoints (without loading locally a complete KG dump). Also, none of them output information about the KG terms.

With respect to ABSTAT, our proposal can be analysed with regard to Thomas Kuhn's six criteria for characterising successful improvements in scientific theories [5]: Generality (the scope of the theory is increased), Simplicity (the theory is less complicated), Explanatory power (the theory gives increased meaning), Fruitfulness (the theory can potentially meet more currently unspecified requirements), Objectivity (the theory provides a more objective shared understanding of the world), and Precision (the theory gives a more precise picture of the world). Computing a TT profile does not require access to the intensional part of the KG (the ontologies), which is mandatory for ABSTAT and other proposals. Moreover, our proposal works by querying online SPARQL endpoints, relieving users of the burden of downloading the KG dump. So, `TTProfiler` provides a support similar to ABSTAT's one, but in less constrained settings, which can be considered to improve the Generality of the tool. Clustering types and terms according to their use in the KG increases the representativity correctness, or Objectivity. Explanatory criteria is hardly applicable for these two tools, but as a TT profile exhibit more information about the KG's content by showing not

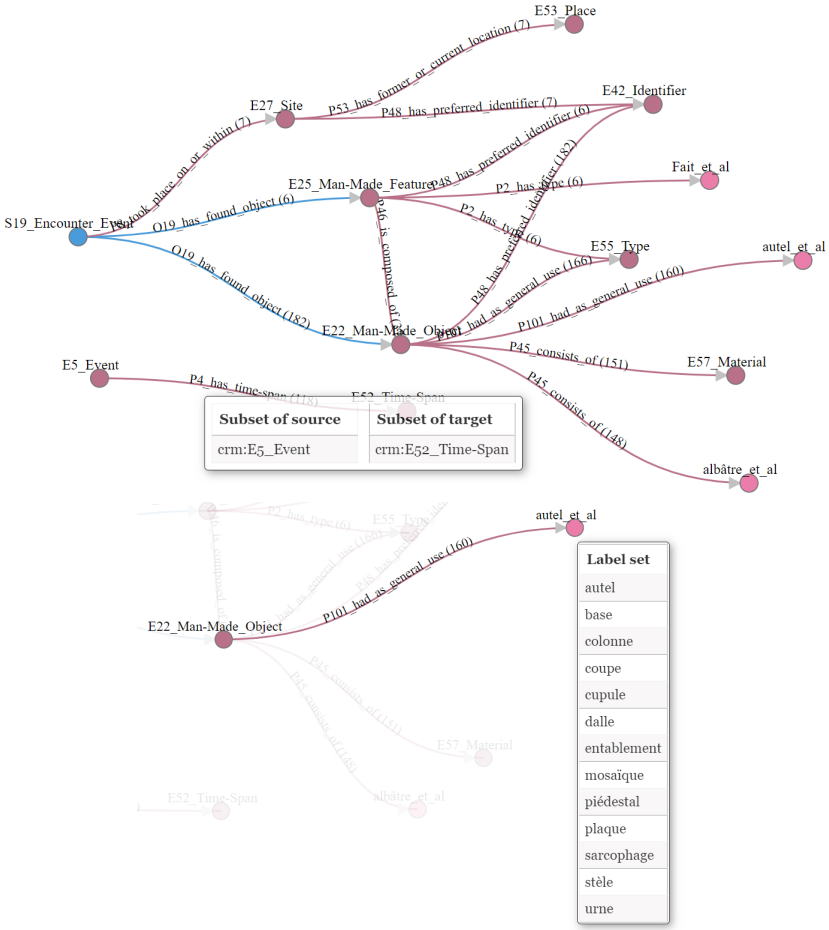

Fig. 4: `Epicherchell`'s profile

only the types and predicates, but also the terms of shared controlled vocabularies that are used, it provides more information about the KG content's meaning. This important point could also be considered to improve the Precision criteria. Regarding Fruitfulness, TT abstract pattern definition allows us to propose a new form of profile, and other ones could still be defined based on TT APs.

As already mentioned, `TTProfiler` is used by OpenArchaeo's KG producers who know which predicates, types and terms should appear in the TT profiles and can detect anomalies in their KG production process. *TT profiles are therefore a support to the automatic generation of KGs.* This interesting aspect is also cited in [4] for summaries' applications. It emerged during our experiments, regarding the use of terms from thesaurus in particular, it revealed several errors in the graphs generated for OpenArchaeo, which could then be corrected. For the

consumers of KGs, TT profiles offer an abstract of the graph content in the same way as ABSTATS' outputs, but with the terms added. It is a well-established practice in humanities and digital libraries to create and use authority lists of terms, i.e. shared controlled vocabularies, and our experiments demonstrated how interesting it is for humanists users to explore terms in TT profile visualisations. But the usefulness of terms goes beyond humanities: categories are first class citizen in Wikipedia, of paramount importance for crowdsourcing stakeholders; folksonomies are also a well known and studied phenomenon. User communities tend to organise themselves to create lists of terms for their needs of descriptions. Being it in a scholarly and structured way as in natural sciences, humanities and libraries, or simply spontaneously like in the social web, the phenomenon must be taken into account when trying to give an idea about a knowledge graph's content.

A TT profile can be used for supporting humans in discoverins KG content to retrieve the information they want. We are currently designing a Web visualisation tool for interactively showing the KG profile, and we plan to provide also an API for applications to query the profile. We are also studying ways of completing the TT profile with information from the ontologies, by extracting the minimal parts of ontologies useful for the profile's types and predicates. Another need demonstrated with our experiments is to build summaries from the profile for huge KGs. We would like to design algorithms for building summaries as sets of k connected nodes, for a given number k.

Acknowledgements: This work is supported by the ANR-18-CE38-0009 ("SESAME"). The authors thank Zilu Yang for her work on the Web visualisation tool.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)
2. Bikakis, A., Hyvönen, E., Jean, S., Markhoff, B., Mosca, A.: Editorial: Special Issue on Semantic Web for Cultural Heritage. Semantic Web 12(2), 163–167 (2021)
3. Boeuf, P.L., Doerr, M., Ore, C., Stead, S., et al.: Definition of the CIDOC Conceptual Reference Model. version 6.2.1. ICOM/CIDOC Documentation Standards Group. CIDOC CRM SIG (2015)
4. Cebiric, S., Goasdoue, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing Semantic Graphs: A survey. The VLDB Journal (2018)
5. Kuhn, T.: Objectivity, Value Judgment, and Theory Choice. University of Chicago Press (1977)
6. Spahiu, B., Porrini, R., Palmonari, M., Rula, A., Maurino, A.: ABSTAT: Ontology-Driven Linked Data Summaries with Pattern Minimalization. In: The Semantic Web - ESWC 2016 Satellite Events, Revised Selected Papers. pp. 381–395. Springer (2016)
7. Troullinou, G., Kondylakis, H., Daskalaki, E., Plexousakis, D.: Ontology understanding without tears: The summarization approach. Semantic Web 8(6), 797–815 (2017)