TTProfiler: Types and Terms Profile Building for Online Cultural Heritage Knowledge Graphs

LAMINE DIOP, EPITA, LRE, France
BÉATRICE MARKHOFF, Université de Tours, LIFAT, France
ARNAUD SOULET, Université de Tours, LIFAT, France

As more and more knowledge graphs (KG) are published on the Web, there is a need for tools that show their content. This implies showing the schema-level patterns instantiated in the graph, but also the terms used to qualify its entities. In this paper, we present a new profiling tool that we call TTprofiler. It shows the predicates that relate *types* in the KG, and also the *terms* present in this KG, because of their paramount importance in most KGs, especially in the Cultural Heritage (CH) domain. We recall the role of terminologies and how they are implemented and used on the Web, we give the algorithm for building a TT profile from an online KG's Endpoint, and we report on experiments performed over a set of Cultural Heritage Web KGs. A tool for visualizing TT profiles is also provided.

CCS Concepts: • **Information systems → Data extraction and integration**; **Web data description languages**.

Additional Key Words and Phrases: CIDOC CRM, Knowledge Graph, Profile Extraction, Terminologies, Visualization

ACM Reference Format:
Lamine DIOP, Béatrice Markhoff, and Arnaud Soulet. 2018. TTProfiler: Types and Terms Profile Building for Online Cultural Heritage Knowledge Graphs. *ACM J. Comput. Cult. Herit.* 37, 4, Article 111 (August 2018), 22 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

It has become widespread in the Cultural Heritage (CH) field to generate Knowledge Graphs from legacy datasets, using one or more ontologies [4]. A knowledge graph (KG) is a dataset in RDF [9], i.e. a set of ($subject$, $predicate$, $object$) triples. CH KGs contribute to the Linked Open Data[1] (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, or KGs, for data analytics requires a clear idea of their content and this is a long-standing challenge. These KGs generally use ontological schemas, composed of classes (types) and predicates. 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. A very common practice is then to look for types and predicates that are instantiated in the graph, and their number of instances.

These types and predicates are "universals" in the sense of metaphysics[17], contrary to the "individuals" described in the dataset (of which universals are abstract representations). Nevertheless, when we try to explore in this way the content of a CH LOD dataset, it quickly becomes apparent that many other universals than types and

[1] https://lod-cloud.net

Authors' addresses: Lamine DIOP, EPITA, LRE, Kremlin-Bicêtre, France, lamine.diop@epita.fr; Béatrice Markhoff, Université de Tours, LIFAT, Blois, France, beatrice.markhoff@univ-tours.fr; Arnaud Soulet, Université de Tours, LIFAT, Blois, France, arnaud.soulet@univ-tours.fr.

XXXX-XXXX/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

predicates exist in those KGs: these are elements of controlled vocabularies, taxonomies, thesauri, more generally terminologies. Those terminological resources are conceived to improve communication among experts in certain domains, and to retrieve information. In the digital world it is well known by database creators that terminologies play an important role for interoperability of the data they store. This is also a W3C recommendation [13]: shared and standardized vocabulary terms (i.e. URIs) must be used to encode data and metadata. The authors of the recommendation state that the benefits of this good practice are: reuse, ease of processing, understanding, trust and interoperability. In the Web, some terminological resources naturally take the form of ontologies formalised using RDFS[2] or OWL[3], others take the form of thesauri using for instance SKOS[4], or, with a more linguistic point of view, the form of lexical resources formalized with an extension of Ontolex-lemon[5], called Terminology Module[6]. *In this article, we are interested in showing the universals used in a KG because they give a clear idea of its structure and content.* In KGs, universals are types and predicates, and elements of a terminological resource that are not types and predicates, but are *associations of a concept to a word, documented somewhere*. We call term the words associated to the concept, and for showing it, we look for the concept's URI (as we look for types and predicates URIs).

In the last ten years, several proposals have been made for helping users in knowing what contains a given KG, by extracting its predicates, the types of entities they link, and some basic statistics. For instance this is what ABSTAT [18] does. Our aim is to generate such a profile of a KG, via its SPARQL Endpoint. ABSTAT does not run on an online SPARQL Endpoint, but it allows us to clearly present what we call a profile: from a given KG, ABSTAT builds a set of (C, P, D) triples with statistics, where C and D are types (classes) and P is a predicate (property). Such a triple is called an Abstract Pattern (AP). Figure 1 shows the four first APs returned by ABSTAT when asking for the predicate `dbo:country` on a dump of DBpedia 2016 in English, using ABSTAT's website[7]. For instance the first AP indicates that there are 560, 532 RDF triples (last column) in this KG for which the predicate `dbo:country` (second column) 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 in DBpedia. In the bottom of Figure 1 we show the Basic Graph Pattern (BGP) able to compute the instances of an ABSTAT AP (n being its frequency, or the number of its instances in the KG, i.e. the last column of the table above). In this BGP, edges labeled with "a" represent the predicate "rdf:type".

Considering again the results returned by ABSTAT in Figure 1, we notice that `dbo:Location` and `schema:Place` 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 (560, 532). More generally, ABSTAT returns thousands of APs just for the predicate `dbo:country` from this dataset, several of them representing the same facts in the KG. If the BGP in Figure 3 (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 a new kind of AP, where the predicate P relates two *sets of types*, as in Figure 3 (a).

Moreover, to the best of our knowledge there is no tool that shows not only the types of the subject and object of a predicate, but also the *terms*, elements of a terminological resource, used to characterise individuals in the KG. For example, it is one thing to see that there are instances of `E22_Human-made_Object` in a graph, but the fact that this graph contains information about coins, or burials, is much more interesting and precise. In KGs that use the CIDOC Conceptual Reference Model [8] (hereafter CIDOC), the reference ontology for Cultural Heritage,

[2]https://www.w3.org/TR/rdf-schema
[3]https://www.w3.org/TR/owl-ref
[4]https://www.w3.org/TR/skos-reference
[5]https://www.w3.org/2016/05/ontolex/
[6]https://www.w3.org/community/ontolex/wiki/Terminology
[7]http://abstat.disco.unimib.it/
[8]cidoc-crm.org/

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

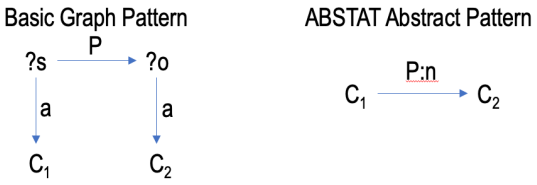

Fig. 1. ABSTAT Abstract Patterns and corresponding Basic Graph Pattern to compute their instances.

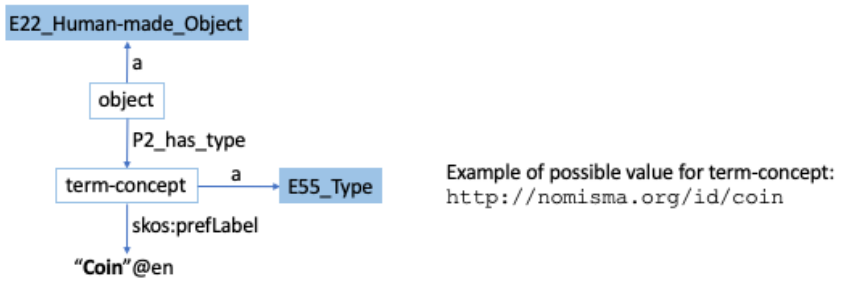

Fig. 2. Example of representing a coin (the *object*) with CIDOC CRM.

this information is denoted by elements of a terminological resource, such as `http://nomisma.org/id/coin` for instance. 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." To this end, the class `crm:E55_Type` is provided as a gateway to these controlled vocabularies. An example using this class is given in Figure 2, and more explanations are provided in Section 2. CIDOC's policy for terminologies 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 are listed and described elsewhere in authority lists, for interoperability purposes. This means that CIDOC-based KGs generally employ various sets of terms, which *provide at least as much meaning and clues on the KG's content as the types and predicates it uses.*

For taking this into account, we define another kind of abstract pattern to show terms used in the graph, shown in Figure 3 (b), where the edge labeled with "prefLabel" represents the situation where the variable ?t is instantiated by a terminological concept and ?l by its label, whatever the property actually used, which may be `rdfs:label`, `skos:prefLabel`, etc. The example in Figure 2 is therefore represented by the Term Abstract Pattern

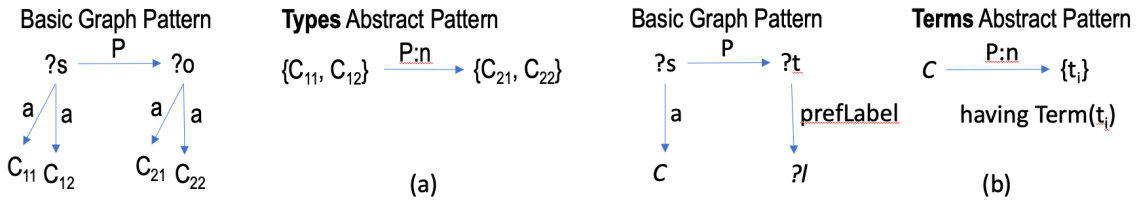

Fig. 3. Types (a) and Terms (b) Abstract Patterns, with their corresponding Basic Graph Pattern (to their left).

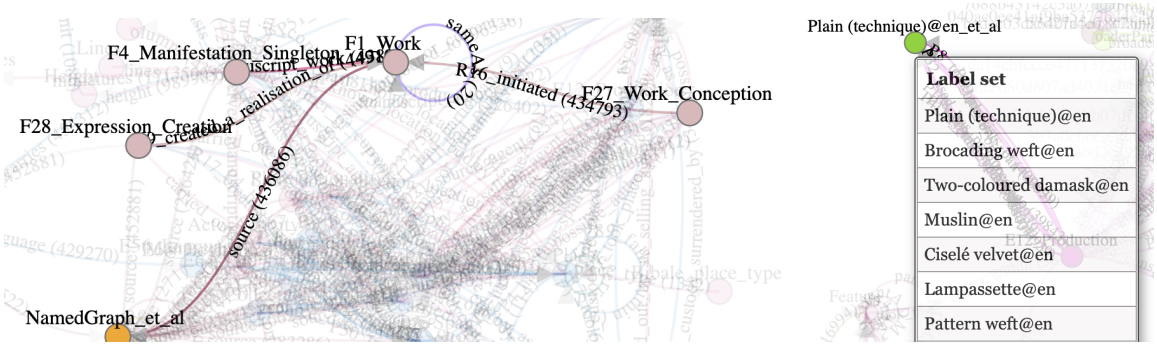

Fig. 4. Excerpts of profiles for MMM (left) and SILKNOW (right), generated by TTProfiler.

where C = `E22_Human-made_Object`, P = `P2_has_type`, t_i = `http://nomisma.org/id/coin` (and $?l$ =**"Coin"**). In this pattern the third item is a set of concepts denoting terms detected by Function *Term*, defined in Section 4, whose implementation depends on how the terminologies are implemented, and how they are used in the KGs.

To sum-up our contribution, we deal with KGs in the LOD as presented in [9], offering an online SPARQL endpoint, which use formally defined existing RDFS or OWL schemas, and contain terminology elements that are defined in existing terminological resources. We consider that the KG may not contain the definition of the schemas and terminological resources it uses, which is the most common situation for Web KGs, so we don't make use of it. We present a program called TTProfiler which builds a set of Types and Terms (TT) Abstract Patterns from an online KG, that we call a TT-profile. It does so by querying the KG's SPARQL endpoint. TTProfiler's code is publicly available[9]. Moreover, some of its results can be visualized online[10]. Figure 4 shows such visualizations for TTprofiles of MMM[11] and SILKNOW[12]. Those knowledge graphs contain millions of RDF triples but their structure and their terms can be explored via their profile. To the best of our knowledge it is the only software that acts by querying a knowledge graph via its SPARQL endpoint and displays its terms, in addition to its types and predicates.

This article extends the one presented at SWODCH 2021 [7] in three ways: firstly, we motivate in Section 2 the particular attention we pay to the terminology elements that appear in KGs, by recalling their utility and the way they are used in the Web. In this way, the notion of term is much more precise and this is propagated in definitions, algorithms and implementations, hence in experimental results too. Secondly, we provide a new

[9]https://github.com/DTTProfiler/DTTProfiler
[10]https://kgsumviz.univ-tours.fr/
[11]Mapping Manuscripts Migration: http://ldf.fi/mmm/sparql
[12]https://data.silknow.org/sparql

section for related work. Thirdly, we present and discuss new experimental results. The rest of this article is organized as follows: Section 2 is dedicated to terms, Section 3 deals with related proposals for giving hints about knowledge graphs' content, in Section 4 we provide definitions and formulate the problem resolved by TTProfiler. In Section 5 we explain its algorithm. We report in Section 6 our experiments on various online Cultural Heritage KGs, and we conclude in Section 7.

2 WHERE ARE THE TERMS WE NEED IN KG PROFILES

When a community wants to share a consensus on how to name universals[13] of its domain, it works on two related questions: what are these universals and which *words* best represent what they are? This defines the concept-terms of the domain, in other words the domain's terminology. In many sciences this is a significant part of the research activities. In this paper, we call *term* an element of a terminology, i.e. a concept and its word(s), or a word and its related concept (cf. Section 2.2, and Definition 4.4). In general, but particularly in the Semantic Web, terminologies are created either by *linguists*, following an onomasiological (the concept is the entry point) or semasiological (the word is the entry point) point of view, or by *ontologists*, or by *information processing professionals* for information retrieval purposes. Terminologies can be lists (controlled vocabularies), or taxonomies, or thesauri (Knowledge Organisation Systems), or ontologies. Their items may be in the intensional part of a KB, taking the form of class and property names (as in the "Terminological Box" of Description logics), or they may be in the extensional part of a KG, taking the form of data, instances of classes (for instance of the `skos:Concept` class). With TTProfiler, when they are classes or properties they are extracted using Types Abstract Patterns, while the in-data terms are extracted using Terms Abstract Patterns shown in Figure 3. In this section, we begin with the term usage policy adopted by CIDOC because it is the basis for our awareness of the fundamental importance of terms, followed by a short survey on terminological resources on the Web, because we think that the Web and its uses bring a new light to the old notion of universals. Then we recall the principles of terminologies, thesauri, and ontologies, in order to clarify what their relationships are.

2.1 The well defined convention adopted by CIDOC

In the Cultural Heritage domain, the community that defined and maintains CIDOC, the CRM SIG, has a clear policy for the use of terminological descriptions in conjunction with the ontology, which we now briefly recall. CIDOC is an ontology designed to *support the semantic interoperability* of digital cultural heritage resources. The use of terminologies is addressed in the introduction of the document that defines it [3], in the section entitled *About Types*[14]. A particular class, `E55_Type`, is intended to group the terminology elements used to characterize and classify the instances of CIDOC classes. The highest class in the subsumption hierarchy, `E1_CRM_Entity`, is the domain of the property `P2_has_type`, whose range is `E55_Type`. Thus, each CIDOC class (except `E59_Primitive_Value`), inherits the `P2_has_type` property, which provides a general mechanism for specializing the classification of instances at any level of detail. This can be done by linking to external sources (thesauri or ontologies). To classify in this way, it is possible to implement the concept either as a subclass (`E55_Type` being used as a kind of Region from a DOLCE point of view [5]), or as an instance of `E55_Type`. According to the foundational principles of CIDOC, a new subclass should only be created if the concept is sufficiently stable and associated with additional properties, otherwise an instance of `E55_Type` must be chosen. In Figure 2 we show for example how the information about a coin is structured in this case. Instances of `E55_Type` can be associated with labels and organized hierarchically (with broader/narrower and part-of properties). Moreover, `E55_Type` is a subclass of `E28_Conceptual_Object`, it therefore also inherits all the properties of this superclass in order to be

[13] In metaphysics universals are what particular things have in common [17].
[14] It is also explained in more detail here: https://www.cidoc-crm.org/FunctionalUnits/taxonomic-discourse

documented. This coherent treatment of terminologies reinforces the capacity of CIDOC to serve as a pivot for the integration of knowledge relating to CH.

2.2 Terminological Resources on the Web

Terminological resources can be found in the form of thesauri or ontologies in the Semantic Web, both of which support semantic interoperability, the former at the data level and the latter at the metadata level. Terminological resources are more massively reused than ontologies designed to represent a knowledge domain, because the consensus required for their reuse is on the terms (and their definition in context), and not on the more complex question of how the domain representation is organized and structured (ontology). It seems to be easier to choose a relevant term from a thesaurus, which most often contains textual definitions of terms, than to understand and reuse an ontology, except for the simplest ones like FOAF and Dublin Core, which are mostly used as terminological resources. Thus, there are many, and in some cases very large, terminological resources on the Web. For example, in the biomedical field, UMLS[15] gathers concepts from several dozen terminologies, MeSH[16] (defined by the US National Library of Medicine) indexes Medline and PubMed article directories, while SNOMED CT[17] gathers several hundred thousand concepts used in clinical environments, in particular for patient records. Similarly, in the environmental domain, AGROVOC[18] gathers more than 38,000 concepts of food, agriculture, fisheries, forestry, etc. with which are associated more than 800,000 terms in 40 languages. In the field of Cultural Heritage, the Backbone thesaurus of DARIAH[19] is an initiative for the aggregation and maintenance of vocabularies built in communities, but it is above all the Getty AAT vocabulary[20] which is used and with which the thesauri produced by projects are aligned. This is the case for example in EUROPEANA[21] or in the ARIADNEplus[22] platform. The latter organizes the possible searches according to three axes When-Where-What: the Getty AAT is used in the What axis, to describe what is searched for, Periodo[23] is used for the When axis and Geonames[24] for the Where axis. It is interesting to note here that, for search axes such as When (historical periods), Where (places) and Who (people, organizations), URIs from authority lists are also used, but they are then *named entities*, which refer to a unique element. On contrary, *terms* are universals in the same way as the types and predicates of an ontology, even when they are instances (of `skos:Concept`, `crm:E55_Type`, or other types). Interestingly, WordNet®[25] is sometimes presented, or used, as a terminology. It has been used for structuring the first versions of YAGO[26], for instance. Basically, it is a large and popular lexical database of English, where words are grouped into synsets, each synset expressing a concept, and synsets are interlinked by means of conceptual-semantic and lexical relations. Last, Wikidata with its more than 100 millions entities[27] increasingly tends to be used as a terminological resource, as many thesauri are now aligned with it, or incorporated in it.

[15] Unified Medical Language System: https://www.nlm.nih.gov/research/umls/index.html
[16] Medical Subject Headings: https://www.ncbi.nlm.nih.gov/mesh
[17] Systematized Nomenclature Of Medicine Clinical Terms: https://www.snomed.org/
[18] urlhttps://www.fao.org/agrovoc/
[19] urlhttps://www.backbonethesaurus.eu/
[20] Art and Architecture Thesaurus: https://www.getty.edu/research/tools/vocabularies/aat/
[21] https://pro.europeana.eu/page/europeana-aat
[22] https://ariadne-infrastructure.eu/Portal/
[23] perio.do
[24] geonames.org
[25] https://wordnet.princeton.edu/
[26] A huge knowledge base of the LOD: https://yago-knowledge.org/getting-started
[27] https://www.wikidata.org/

2.3 Terminology Definition, and Implementations on the Web

A terminology is particularly useful for filling in database fields in a consistent manner. In this sense, terms are *data*. ISO/TC 37's ISO 1087:2019 defines a terminology as a set of designations used in a specialty language, where a designation represents a *concept* by a *sign* that denotes it. ISO/TC 37 is also the originator of the TMF (Terminological Mark-up Framework) and LMF (Lexical Mark-up Framework) standards that inspired the OntoLex-lemon[28] ontology (representation of morpho-syntactic properties of lexical entries and their meanings) and its extension for terminology (in the process of being defined[29]), which is dedicated to documenting information about terms. This ontology allows to clearly represent the interface between syntax (sign) and semantics using Class *LexicalSense*, which can be linked to an ontology in which the concept is described. This question of linking lexical forms and concepts for the description of terms is also the subject of a proposal called onto-terminology [16]. But these are not very commonly used implementations for terminologies on the Web. Rather, some of them take the form of ontologies, like the Dublin Core and DCTerms[30], or the quality regions in DOLCE [5], while most of them are realized with SKOS[31]. It is an ontology for defining thesauri, taxonomies, classification schemes or subject heading systems, used in documentary systems for indexing and information retrieval purposes. Moreover, some terminologies on the Web are implemented with other Knowledge Organisation System ontologies than SKOS, which also contain classes for representing terms, for instance Schema.org[32] has classes `schema:DefinedTermSet` and `schema:DefinedTerm`, and CIDOC has `crm:E55_Type`, that we already presented.

2.4 Thesauri Definition, and Purposes

ISO 25964-1 defines a thesaurus as a controlled and structured vocabulary which concepts are represented by terms, relationships between concepts are made explicit, and preferred terms are completed with synonyms. A concept is a unit of thought and a term is a word or phrase used to label a concept (ISO 25964-1 sections 2.11 Concept and 2.61 Term). These definitions are similar to those of a terminology. The difference between thesauri and terminologies, however, lies in their purpose. The aim of defining terminologies is to reach a consensus on the designations used in a domain, *for semantic interoperability*. Whereas the main objective of thesauri is *to index and retrieve* elements according to their content. The thesaurus is then used as an access structure: the declarations of synonymy between terms on the one hand, and the hierarchical relationship between concepts on the other hand, allow the information retrieval system to widen or restrict the queries. For this purpose, the hierarchy used in a thesaurus covers the subsumption relation, the partition relation, and sometimes also the instance relation, merged into a single hierarchical relation: *Concept A is broader than Concept B if in any search for A, articles dealing with B should be returned.* This usage-based hierarchical relation does not correspond to any mathematical logic. Large thesauri are organized into facets, which group together hierarchies of concepts to facilitate information retrieval. As noted by [12], the structure of thesauri can not be used for more general reasoning than information retrieval, unlike ontologies.

2.5 Ontologies, versus Thesauri

We will not define here what an ontology is in the Web, as the authors of [12] do, but it is important to clarify its differences with respect to a thesaurus. Formal and consensual model of a shared conceptualization of a domain of knowledge, it consists of intensional (TBox) and extensional (ABox, gathering instances) descriptions [2]. It includes a set of entities (classes, or types), roles (properties, or predicates), constructors, and axioms to describe

[28] urlhttps://www.w3.org/2016/05/ontolex/
[29] urlhttps://www.w3.org/community/ontolex/wiki/Terminology
[30] See https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
[31] See https://www.w3.org/TR/skos-reference/
[32] See https://schema.org/

them. An entity is seen as a class that has instances, the subsumption link between entities being that if A subsumes B then any instance of B is also an instance of A. Roles are described by their domain and range and there may be a subsumption link between them. According to the needs, other constraints can be specified on entities and roles. *All these declarations are automatically exploitable by reasoners, which is not the case for thesauri.* It is possible to represent terminologies in ontologies as topic spaces, for instance with DOLCE regions [5] as done in *ontopic*[33], an ontology for modeling topics, subjects, or themes of something. This makes it possible to benefit directly from the reasoner: for instance asking for resources dealing with places will return resources dealing with all the kinds of places subsumed by the class `Place`. We recognize here the vocation of a thesaurus, so an ontology covers the thesaurus purposes. For this reason, the question of transforming the knowledge contained in a thesaurus into an ontology, in order to exploit it automatically by a reasoner, has been addressed in many works. Among others, [11] shows why it is not simple by any means. Firstly, the broader-narrower relationship in a thesaurus covers both the subsumption relation, the partition relation and the instantiating one. Secondly, the generic associative relation often present in a thesaurus is also complex to transpose automatically into an ontology. Moreover, it must be decided which concepts of the thesaurus must become classes and which become data (instances). Nevertheless, there exist versions of SNOMED CT and AGROVOC in the form of OWL ontologies.

This overview of the presence of terms in LOD demonstrates the usefulness of showing them in a KG profile, because they clearly inform about its content. This also gives an idea of the variety of possible representations for terms in LOD, which makes their extraction non-obvious. This is why we use Function *Term*, defined in Section 4, to subsume the various concrete implementations of term detection.

3 RELATED WORK ON KNOWLEDGE GRAPH PROFILING

To profile a knowledge graph, it is of course possible to use a generic graph profiling method. These methods are interesting, but they tend to disregard the semantics of the relations and they ignore the distinction between the schema and the instances. Yet, these two dimensions are exactly what we seek to capture in our profiles. For this reason, generic graph profiling methods are out the scope of this related work section. As explained earlier, we want to construct a knowledge graph profile that captures the main usage patterns of the schema across its instances. First, a task close to knowledge graph profiling is schema discovery [10] from instances. This task aims at building representative classes from the data. These methods therefore appear rather as a preliminary task to knowledge graph profiling, as in our setting types are defined in ontologies and are used in the KGs we analyze. Second, several approaches in the literature propose to summarise a knowledge graph based on the concept of quotient graph [6, 8]. The key idea of quotient graphs is to define equivalent classes among the original graph nodes and to assign a representative node for each class. Interestingly, it is possible to answer some queries from the summary instead of considering the original knowledge graph. In the same way, the characteristic sets [15], which are other structures to represent node classes, provide an accurate cardinality estimation for RDF queries with multiple joins. [10] provides a list of potential uses of such discovered structures in KGs exploitation. Finally, many KG summarization techniques select a small number of nodes and relations benefiting from user interaction [19, 22] or centrality measures [21, 23]. For instance, RDF digest [20] is a method that selects several types using a centrality metric. Then, relations with intermediate types are added in order to link the initially selected types. On the contrary, profiling aims at representing all the information present in the knowledge graph. Furthermore, to the best of our knowledge, all the methods based on schema discovery or summarization do not extract *patterns* describing the main relations between types, but rather focus on classifying *nodes*. We found only one exception: ABSTAT. ABSTAT [1, 18] builds a summary by identifying the *main relationships between types*, called Abstract Patterns (cf. Figure 1). Compared to TTProfiller, this approach takes into account the type hierarchy to remove from the resulting profile, which is an intermediate result, some subsumption redundancies.

[33] http://www.ontologydesignpatterns.org/ont/dul/ontopic.owl

This leads to a summary. However, redundancies remain because many patterns are incomparable. Also, all facts of the KG are represented in ABSTAT's profiles because entities that are not explicitly declared as instances of some type are grouped in `owl:Thing`, while TTProfiller ignores these entities for now.

The main weakness of all these approaches proposed in the literature, compared to our aim of highlighting the topics of a KG, is that they all ignore the terms present in the KG, while terms are often essential to understand its organization (especially in the case of Cultural Heritage where the use of terms is omnipresent). Of course, it would still be possible to apply one of these methods by ignoring terms (scenario 1). We could also apply one of these methods, search for terms, and add a type per term (scenario 2). *Scenario 1:* As explained in Introduction and in more detail in Section 2, terms play an important role in general, and in Cultural Heritage knowledge graphs in particular, especially for those exploiting CIDOC because CIDOC classes and properties are abstract ones for the purpose of interoperability, and can be further specified by using terminologies. Ignoring the terms for these KGs is ignoring the real topics of their contents. *Scenario 2:* Adding a type per term is not straightforward. First, since the terms are not used as types in the KG, the basic graph patterns devised for profiling cannot be used for terms. Second, a large number of terms (which is the case in most KGs) would lead to an explosion of nodes and edges in the profile, with important redundancies.

Contrary to these approaches, we take into account the terms which are very informative entities to describe the content of a knowledge graph. Taking terms into account has an impact on the design of profiles as we propose to build virtual nodes allowing to gather several types or terms. This guarantees a compact form of the output and brings out common relationships. Finally, we propose a method that relies on SPARQL queries that are simple enough to directly querying public endpoints. This aspect is very important to avoid having to download and process huge dumps [1, 8], and to propose profiles that are always up-to-date with respect to the available data.

4 DEFINITIONS AND PROBLEM FORMULATION

Notations defined in this section are summarized in Table 1. We use Description Logics (DL) [2] notion of ABox for defining our problem: a knowledge base (KB) $\mathcal{K}$ is composed of a TBox $\mathcal{T}$ (names and assertions about concepts and roles, respectively called types and predicates in what follows) and a ABox $\mathcal{A}$ (assertions about individuals, called entities and facts). For instance DBpedia[34] 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 we find in DBpedia `Person` (`dbr:Ringo_Madlingozi`), `skos:Concept` (`Category:1964`) and `dct:subject` (`dbr:Ringo_Madlingozi`, `Category:1964`). In the Web all this is written with triples $(subject, predicate, object)$, for instance the two last examples correspond to the two following RDF triples: (`Category:1964`, `rdf:type`, `skos:Concept`) and (`dbr:Ringo_Madlingozi`, `dct:subject`, `Category:1964`). Like most of KGs on the Web, DBpedia does not use only its own ontology but many other ones, as SKOS and DCTerm in the previous examples. This means that entities and facts in Web KGs instantiate types and predicates coming from many different ontologies.

What are the Knowledge Graphs we want to analyze? The KGs we analyze are ABoxes, which in the Web of data are in general far bigger than TBoxes. We work with the asserted KG that is queriable via its SPARQL Endpoint. By default, SPARQL endpoints do not perform entailments. So in this paper, *what we call KG is a ABox (i) whose entities and facts instantiate types and predicates coming from many different ontologies, and (ii) that is not saturated*

[34]The well known hub of the LOD that mirrors for programs the content of Wikipedia; its SPARQL human interface is: `https://dbpedia.org/sparql`

Table 1. Notations

Symbol	Denotes
$?s$	variable representing subjects
$?o$	variable representing objects
t	term
P	predicate
C	type of an entity (class)
$\mathcal{A}$	ABox, assertions about individuals, called entities and facts
$\mathcal{T}$	TBox, names and assertions about types and predicates
$Term(\mathcal{A})$	function that returns the set of terms of $\mathcal{A}$
$\mathcal{C}$	set of types (classes) appearing together as subject or object into a TT abstract pattern
$\mathcal{D}$	set of concepts (terms) appearing together as object into a TT abstract pattern $(C, P, \mathcal{D})$
$\mathbb{P}$	TT profile
$\mathcal{K}$	Knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
$\omega((C, P, \mathcal{D}))$	weight of the TT abstract pattern $(C, P, \mathcal{D})$, frequency of $(C, P, \mathcal{D})$ in $\mathcal{A}$
$\mathcal{AP}$	TT abstract pattern
$\mathcal{C}(\mathcal{K})$	set of types, i.e. entities appearing as object of `rdf:type` into the knowledge base $\mathcal{K}$
$\mathcal{BP}$	basic patterns
PV	profile visualization structure

by applying a reasoner. Exploiting the ontologies and thesauri that are used in the KG is out of the scope of this work, but will be considered as future work. When publicly available, they can be used latter on, for enriching the information already present in the profile. There may also 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.

What we extract from these Knowledge Graphs. We put in evidence all the types and predicates used in the ABox, whatever the ontologies they belong to. Web KGs frequently use several ontologies, so we do not limit ourselves to only one given ontology. We also want to show the terms used in the KG, whether they are defined in the graph itself or come from external resources. Remember that in this paper we call term an element of a terminology, as defined in ISO 1087:2019. We saw in Section 2 that, in the Web of data, such an element may be implemented either as a TBox element or as a ABox element. In the first case, if the type characterizes an entity in the KG then it will appear in the profile. For the second case, as terms are implemented in many different ways in Web KGs, we define a generic function called $Term$, implemented by a SPARQL query. For example, it may look for `skos:Concept` or `crm:E55_Type` instances, or for subjects of `skos:prefLabel` or objects of `crm:P2_has_type`, or for declared prefixes that correspond to some known thesauri, or any combination of these features. We consider three implementations of Function $Term$ in Section 6 (Figure 9), among those many possibilities deriving from Section 2. Hence, the following definition does not state that Function $Term$ returns *all* terms of a KG.

Definition 4.1 (Function Term). Given a ABox $\mathcal{A}$, Function $Term$ extracts from $\mathcal{A}$ URIs of concepts defined in a terminology, which are not used as types in $\mathcal{A}$.

Definition 4.2 (Type). Given a ABox $\mathcal{A}$, a type used in $\mathcal{A}$ is an entity t that appears as object of Predicate `rdf:type` in $\mathcal{A}$, i.e. $(s, \texttt{rdf:type}, t) \in \mathcal{A}$.

Definition 4.3 (Predicate). Given a ABox $\mathcal{A}$, a predicate used in $\mathcal{A}$ is a resource P that appears as predicate in $\mathcal{A}$, i.e. $(s, P, o) \in \mathcal{A}$.

Definition 4.4 (Term). Given a ABox $\mathcal{A}$, a term appearing in $\mathcal{A}$ is an entity t such that $t \in Term(\mathcal{A})$.

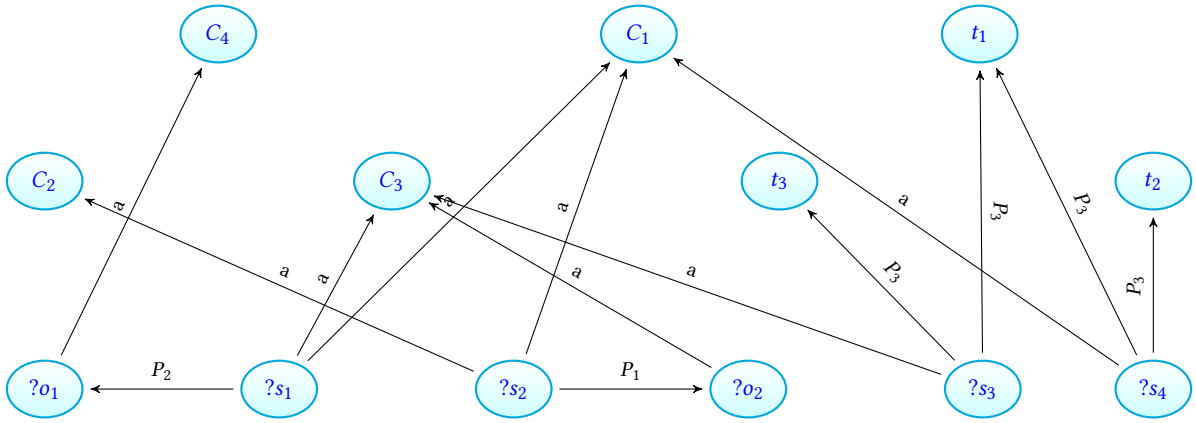

Fig. 5. Example of a knowledge graph

Example 4.5. Figure 5 shows a toy example of a knowledge graph $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ in order to illustrate our approach where the set of types is $\{C_1, C_2, C_3, C_4\}$, the set of predicates is $\{a$ (for `rdf:type`$), P_1, P_2, P_3\}$ and the set of terms is $Term(\mathcal{A}) = \{t_1, t_2, t_3\}$. The other entities in $\mathcal{K}$ are denoted either by $?s_i$ or by $?o_i$. For instance, the entity denoted by `object` in Figure 2 may be represented by $?s_4$, with C_1 standing for `E22 Human-made Object`, P_3 standing for `P2 has type` and t_1 standing for the entity denoted by `term-concept`.

The algorithm presented in this paper builds a *profile* of the KG queried via its SPARQL endpoint. We call $\mathcal{A}$ this KG as it is a ABox. The resulting profile is composed of TT AP (Types and Terms Abstract Patterns), which are triples whose subjects and objects are sets. This is defined in Definition 4.6. As discussed in Introduction, APs in [18] are triples (C, P, D), where C and D are types and P is a predicate: we call them *basic APs*. TT APs generalise basic APs in two ways: first, objects can be either types or terms. Second, both subjects and objects are sets (either set of types or set of terms), as illustrated in Figure 3.

Definition 4.6 (TT Abstract Pattern, and represented facts). Given a 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 used in $\mathcal{A}$, P is a predicate used in $\mathcal{A}$, and $\mathcal{D}$ is either a set of types used in $\mathcal{A}$ or a set of terms appearing in $\mathcal{A}$. A TT abstract pattern $(\mathcal{C}, P, \mathcal{D})$ represents the fact $P(a, b)$ of $\mathcal{A}$ iff:

- The entity a occurs in $\mathcal{A}$ as an instance of each type in $\mathcal{C}$ (i.e., $C_i(a) \in \mathcal{A}$ for $C_i \in \mathcal{C}$), and there is no other type in $\mathcal{A}$ of which a occurs as an instance, and
- The entity b occurs in $\mathcal{A}$ as an instance of each type in $\mathcal{D}$ (i.e., $D_i(b) \in \mathcal{A}$ for $D_i \in \mathcal{D}$) and there is no other type in $\mathcal{A}$ of which b occurs as an instance, or the entity b is a term and $b \in \mathcal{D}$ (i.e., $b \in (Term(\mathcal{A}) \ \& \ \mathcal{D})$).

This definition can be adapted to other cases, depending on what is considered as input. 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 [18]. For terms, one could also use some definitions in their respective thesaurus. But in our context, as already stated, we do not access external resources. Contrary to [18], if a or b have several types asserted in $\mathcal{A}$ (whether or not linked in $\mathcal{T}$ by a subsumption) then by Definition 4.6 the fact $P(a, b)$ is represented by only one AP. Also contrary to [18], a fact $P(a, b)$ having no type asserted for a, or having neither a type asserted for b nor any clue allowing to detect that b is a term, does not raise any AP.

Example 4.7. Figure 6 depicts the 4 TT abstract patterns stemming for the knowledge graph depicted by Figure 5: $(\{C_1, C_2\}, P_1, \{C_3\})$ (in red), $(\{C_1, C_3\}, P_2, \{C_4\})$ (in green), $(\{C_1\}, P_3, \{t_1, t_2\})$ (in blue) and $(\{C_3\}, P_3, \{t_1, t_3\})$ (in gray). For instance, $(\{C_1, C_2\}, P_1, \{C_3\})$ means that the subjects of types C_1 and C_2 are related by P_1 to the objects

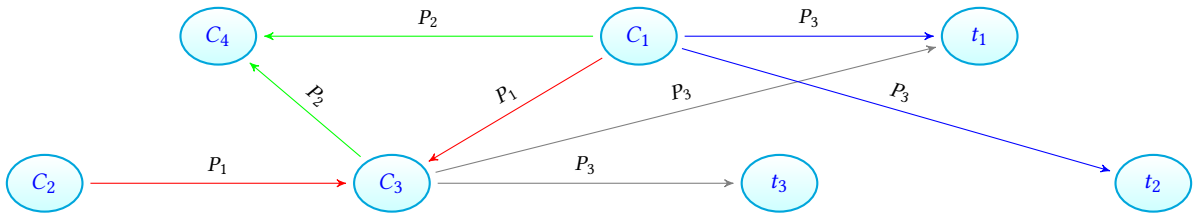

Fig. 6. Abstract of the knowledge graph in Figure 5

of type C_3. Interestingly, some TT abstract patterns group together some terms used in the same way (e.g., $\{t_1, t_3\}$). As illustration, in Figure 4 the set of terms denoted by "Plain (technique)@en_et_al" sums up some techniques of silk weaving like brocading weft or two-coloured damask.

Given the set of APs generated from an ABox $\mathcal{A}$ according to Definition 4.6, we can associate statistics with those patterns, leading to the following definition of a *TT profile*:

Definition 4.8 (TT Profile). Given an ABox $\mathcal{A}$, a TT profile $\mathbb{P}$ of $\mathcal{A}$ is a set of pairs $((C, P, \mathcal{D}), w)$ such that $(C, P, \mathcal{D})$ is a TT AP generated from $\mathcal{A}$, and w is a statistic value describing $(C, P, \mathcal{D})$.

There are many ways to define interesting statistics of a KG. 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 term b appearing in $\mathcal{A}$, and so on. 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 $(C, P, \mathcal{D})$ its frequency in $\mathcal{A}$.

Definition 4.9 (Weight of a TT Abstract Pattern). The weight of the TT abstract pattern $(C, P, \mathcal{D})$, denoted $\omega((C, P, \mathcal{D}))$, is the function that associates with $(C, P, \mathcal{D})$ its frequency in $\mathcal{A}$. $\omega((C, P, \mathcal{D})) = |\{P(a, b), P(a, b) \in \mathcal{A}$ and $P(a, b)$ is represented by $(C, P, \mathcal{D})$ according to Definition 4.6$\}|$.

Last, to reduce the number of nodes to be displayed in the *profile visualization*, we perform a little optimisation by grouping the sets of types and the sets of terms, in such a way that each type, or term, appears in only one set (node) of the profile.

Example 4.10. If we have in a TT profile $\mathbb{P}$ the TT APs $\mathcal{AP}_1 = (\{C_1, C_2\}, P_1, \{C_3\}))$, $\mathcal{AP}_2 = (\{C_1, C_3\}, P_2, \{C_4\})$, $\mathcal{AP}_3 = (\{C_1\}, P_3, \{t_1, t_2\})$ and $\mathcal{AP}_4 = (\{C_3\}, P_3, \{t_1, t_3\})$, with $\omega(\mathcal{AP}_1)$ =20, $\omega(\mathcal{AP}_2)$ =18, $\omega(\mathcal{AP}_3)$ =100 and $\omega(\mathcal{AP}_4)$ =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 7[35].

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. In the profile visualization (cf. Section 6.2), maximal nodes are represented by the name of one of their types or terms followed by *et_al*, and the others are shown on demand. In the same way as in Figure 7, edges are annotated with the corresponding AP and its respective weight.

Problem formulation. **Given the assertional part $\mathcal{A}$ of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, how to efficiently generate and visualize a TT profile $\mathbb{P} = \{((C, P, \mathcal{D}), w)\}$ of $\mathcal{A}$, where w denotes the weight of abstract patterns and C and $\mathcal{D}$ denote maximal sets)?**

[35]The notation $((\{C_1, C_2\}, P_1, \{C_3\}), 20)$ associates the AP $(\{C_1, C_2\}, P_1, \{C_3\})$ to its weight: 20.

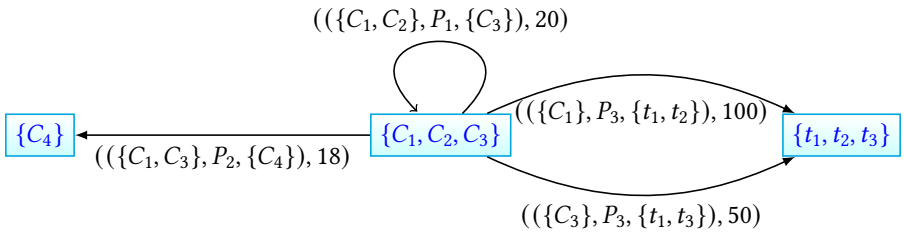

Fig. 7. Graph with maximal sets

5 TTProfiler ALGORITHM

In this section, we first present the general workflow of our program, that we call TTProfiler, and then we focus on the three-step procedure to build a TT profile, formalized in Algorithm 1.

5.1 Workflow of TTProfiler

Figure 8 presents the workflow of TTProfiler. Given a knowledge base $\mathcal{K}$, it first extracts the set of concepts $\mathcal{C}(\mathcal{K}) = \{C_1, \cdots, C_n\}$ and that of predicates $\mathcal{P} = \{P_1, \cdots, P_m\}$ with *getClasses* and *getPredicates* respectively. After that, it computes the *basic patterns* (i.e. ABSTAT abstract patterns as depicted in Figure 1) that can be obtained from the combination of the two previous extracted sets: $\mathcal{BP} = \{(C_{i_1}, P_j, C_{i_2}) : (C_{i_1}, C_{i_2} \in \mathcal{C}(\mathcal{K})) \wedge (P_j \in \mathcal{P})\}$. These extractions and computations rely on SPARQL queries. All computations requiring a SPARQL query are represented by dotted brown lines in Figure 8. The following step consists in computing the weight w of each pattern $(C_{i_1}, P_j, C_{i_2}) \in \mathcal{BP}$. Then, it collects the terms of the knowledge base with the *getTerms* method in Figure 8, such that for each term t we have a triple (C, P, t) with $C \in \mathcal{C}(\mathcal{K})$ and $P \in \mathcal{P}$. It also computes the weight for each of these triples. From the union of the sets of weighted concept-based basic abstract patterns $((C_{i_1}, P_j, C_{i_2}), w)$ and weighted term-based basic abstract patterns $((C, P, t), w')$, we compute the corresponding *TT Abstract Patterns* that are contained in the resulting TT profile and the structure to visualize it. This is detailed in Algorithm 1. Notice that we also collect the *data properties* that qualify the instances of each profile node, in order to show them on demand.

5.2 Algorithm of TTProfiler

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

Step 1: Basic Abstract Patterns and Statistics Recovery. We mine 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 basic abstract 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 in $\mathcal{A}$ (cf. Definition 4.6). Note that in the case where b is a term, then all elements of D are also terms. In other words, a type and a term are not grouped together.

Step 2: Profile Computing. To fit Definitions 4.6 and 4.8, 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

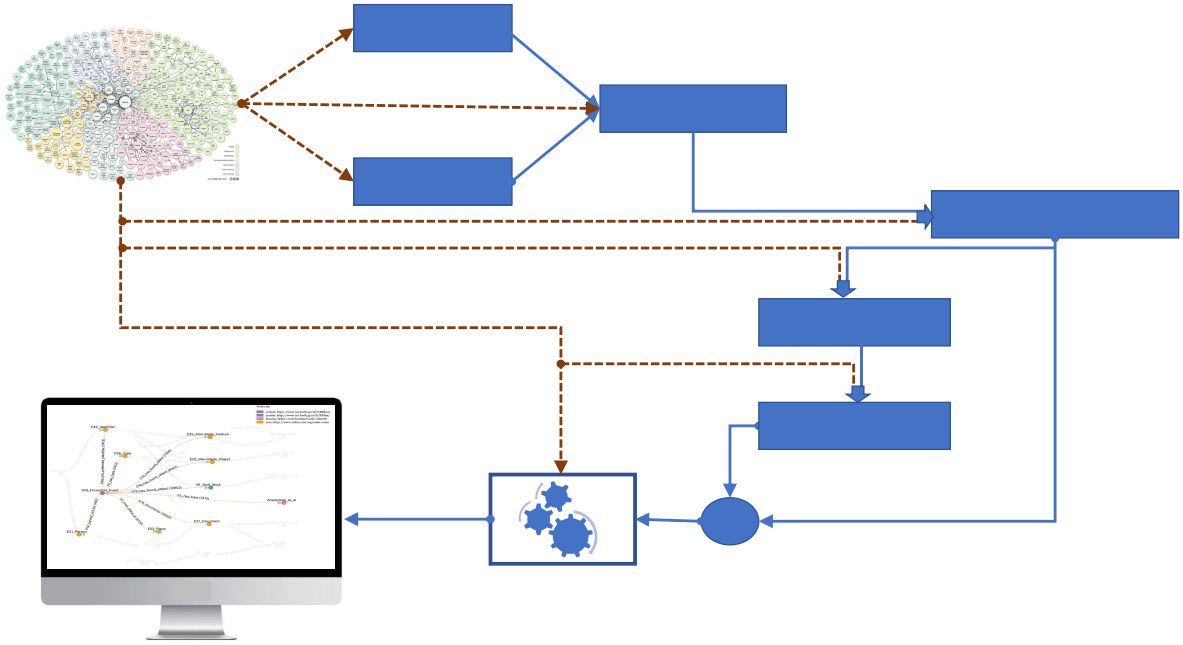

Fig. 8. The workflow of TTProfiler

of a node in φ consists in grouping its elements with other nodes containing them, as explained in Section 4 (cf. the example illustrated in Figure 7).

Step 3: Profile Visualization 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 Visualization 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.

6 EXPERIMENTS ON CULTURAL HERITAGE SPARQL ENDPOINTS

We first explain how Function *Term* can be implemented, and how it is implemented in TTProfiler. Next, we present the knowledge graphs we used for experiments, we show some features of their profiles and an example of profile visualization. We finally report the running times.

6.1 Implementation of Function Term

Our aim is to show in KG profiles terms that are universals, in the same way as classes and properties, in cases where those terms are data. As shown in Section 2, there are several ways to use terms as data in KGs, so Function *Term* has several possible implementations. It is therefore necessary to experiment the following ones until founding the best one for each graph.

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 visualization structure PV
//**Step 1:** ***BASIC*** *abstract patterns extraction from* $\mathcal{A}$ *and statistics computation*
Let $R = \{((C, P, D), w) / (\exists P(a, b) \in \mathcal{A}) \wedge C(a) \in \mathcal{A} \wedge (D(b) \in \mathcal{A} \vee b \in Term(\mathcal{A})))\}$ 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 $(D_1 \in Term(\mathcal{A})) \wedge (D_2 \in Term(\mathcal{A})) \wedge (C_1 = C_2)$ **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 visualization structure*
$PV \leftarrow \emptyset$
for $((\mathcal{C}, P, \mathcal{D}), w) \in \mathbb{P}$ **do**
  $\mathcal{X} \leftarrow maxNode(\mathcal{C}, \varphi)$
  $\mathcal{Y} \leftarrow maxNode(\mathcal{D}, \varphi)$
  $PV \leftarrow PV \cup (\mathcal{X}, ((\mathcal{C}, P, \mathcal{D}), w), \mathcal{Y})$
return ($\mathbb{P}$, PV)
// $add(\varphi, C)$ returns the set of maximal nodes φ having incorporated C
// $maxNode(C, \varphi)$ returns the maximal node that contains C

Dedicated Class. Sometimes, concepts denoting terms are explicitly declared in the analyzed graph as instances of a class that is known to represent terminological resources, for instance `skos:Concept` in Figure 9. Other dedicated classes may be `skos:Collection`, `schema:DefinedTerm` for Schema.org, `ontolex:LexicalConcept` for OntoLex-lemon Terminology or `crm:E55_Type` for CIDOC. We saw indeed that for knowledge graphs using CIDOC and following its recommendations, the instances of `crm:E55_Type` (and its subclasses) must be elements of a terminology. This clue is safe. The class `skos:Concept` alone allows to cover many terms by its frequent use. Unfortunately, the knowledge graph queried via its SPARQL endpoint does not often contain this statement, which is present only in the remote thesaurus. Moreover, using this clue requires to know well the principles of the ontologies used in the KG (for example for CIDOC to know that the class `crm:E55_Type` has for instances elements of terminologies). Last, it supposes that these ontologies are correctly used in the KG.

Dedicated Property. The subjects or objects of some properties have a chance of being concepts denoting terms, for instance the subjects of `skos:prefLabel` in Figure 9, or the objects of `crm:P2_has_type`. We first thought that `skos:prefLabel` would be a strong clue in the same way that `rdfs:type` identifies surely a class instance. In practice, however, this is not at all the case because, even if this property is widely adopted, there is no convention

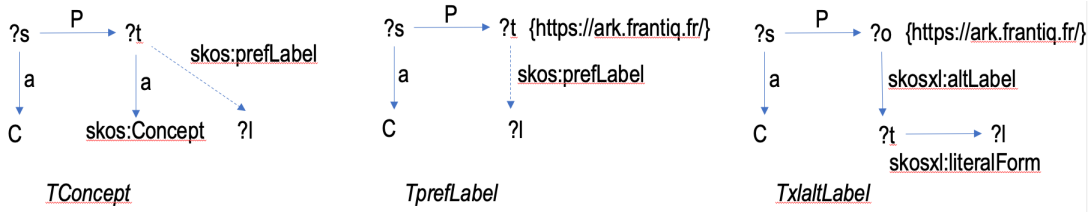

Fig. 9. Three implementations of Function Term.

for its use: its definition does not constrain its domain and there is no clear good practice for using it. As a result, the precision of this hint with this property is low. The recall is potentially high, however it may happen that, rather than a `skos:prefLabel`, it is a `skosxl:prefLabel` that is used, when the graph creator preferred to associate a lexical entity to the concept, rather than directly a character string. Other properties relevant to a terminology description can be tested, such as other SKOS properties, those of SKOS-XL, `ontolex:isEvokedBy` for OntoLex-lemon, $inDefinedTermSet$ and $termCode$ (that have `schema:DefinedTerm` as domain), or `P127_has_broader_term` and `P150_defines_typical_parts_of` for CIDOC. But, as for the dedicated class clue, these properties are rarely present in the analyzed graph, but rather in the graph where the terminology is defined.

Dedicated URI. A naive but efficient method is to detect URI prefixes corresponding to known thesauri, for instance https://ark.frantiq.fr/ in Figure 9. For example, all URIs beginning with http://vocab.getty.edu/aat/ belong to the thesaurus Getty AAT[36]. The accuracy of this approach is obviously very high, but its recall depends on whether the targeted knowledge graph uses known thesauri or not. Moreover, it does not allow the discovery of new thesauri and is therefore not very suitable for unknown knowledge graphs.

In practice, none of the previous methods can work perfectly on all Web KGs, because of the variety of terminology implementations and uses on the Web, presented in Section 2. Thus, Function $Term$ must be tailored for each KG. This can be done based on the KG's metadata and some simple tests. Dedicated properties or classes are very useful, in particular because they provide clues that help detecting vocabulary prefixes, be they internal or external to the KG. These vocabulary prefixes help identifying terms via the URI prefix of their associated concept. But, it is tricky to exploit dedicated properties or classes if they are used for anything other than terminology or thesaurus elements, which is quite frequent. So, most often Function $Term$ has to be implemented using a combination of the previous methods. We show in Figure 9 the three combinations used for computing the profiles of the thirteen graphs that we analyzed. The corresponding SPARQL queries can be found in the Git repository referenced by footnote 9.

6.2 Knowledge Graphs and their TT Profiles

We first tested TTProfiler on archaeological KGs grouped in the semantic Web platform OpenArchaeo[37] [14]. 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 OpenArchaeo's KGs, so the visual query tool that OpenArchaeo provides, which is a special case of Sparnatural[38], may be complemented by the display of TT profiles, to show what can be asked from those

[36]It is of course necessary that all concepts of the thesaurus share the same prefix and that this prefix designates only concepts denoting terms. This is a problem for thesauri like SNOMED CT, for instance
[37]http://openarchaeo.huma-num.fr/explorateur/home
[38]http://sparnatural.eu/

graphs. In addition, the producers of these graphs use the TT profiles to inspect the results of their KG automatic generation workflow, which is based on mappings expressed with tools like Ontop[39] and X3ML[40]. Those KG producers know which predicates, types and terms they want to appear in their graphs, and they use the TT profiles to easily detect anomalies in the generated graphs, which denote anomalies in their mappings.

Besides the KGs in OpenArchaeo, we looked for other graphs that use CIDOC and offer a SPARQL endpoint for programs. Of those found, many are not always online, and many do not answer to counting SPARQL queries of Step 1. We present in Table 2 thirteen graphs that were regurlarly capable of answering the Step 1 queries during May-June-July, 2022. Seven of them are from OpenArchaeo (`Kition`, `Iceramm`, `Arsol`, `Epicherchell`, `Outagr`, `Rita` and `Chronique`). They are rather small (at most 670 000 RDF triples) and contain various datasets like recordings of archaeological excavations, a corpus of antique inscriptions from Cesarea in Mauretania, an inventory of Roman hydraulic works in Northern Italy, etc. The other KGs that we report are: `Smithsonian`[41] that contains data about museum collections (sculpture, painting, photography), `Silknow`[42] that is about European silk heritage, `Culturaitalia`[43] that contains italian museum collections descriptions, and `Doremus`[44] that groups datasets about music, coming from BnF, Radio France and Philharmonie de Paris. The two last datasets have been addressed to show that TTProfiler scales up to quite big graphs, but their TT profiles are still too large to be clearly shown and require more sophisticated means to be explored. More precisely, `Beniculturali`[45] contains datasets from various italian cultural sources: database of the Places of Culture; records of Archives and Libraries; database of the Catalogue of Cultural Heritage; and other documentary and photographic databases. Lastly, `DBpedia` is considered here only for comparison with ABSTAT. All these graphs are of different designs, use different terminologies in various languages, and are of various sizes. For those like `Doremus` and `DBpedia` which contains multilingual terms, we choose only English for our experiments. All these KGs except `DBpedia` use types (classes) of CIDOC and some of its extensions (CRMsci, CRMarch, CRMba, ...). `Doremus`, `Silknow` and `Beniculturali` use its so-called Erlangen implementation. `Beniculturali` and `DBpedia`[46] use types of many ontologies. Concerning the predicates, again all KGs except `DBpedia` use those of CIDOC and its extensions, and sometimes predicates of other ontologies too. Regarding the vocabularies, OpenArchaeo's graphs use the PACTOLS[47], and `Smithsonian`, `Doremus`, `Culturaitalia`, `Beniculturali` and `Silknow` use internally defined vocabularies and external ones, for instance the Getty AAT[48]. When thesauri are used, the number of terms is always larger than that of types.

All the profiles of the graphs presented in this paper can be visualized and explored online through a web application (cf. footnote 10), implemented in Javascript. More profiles are provided in this web application, for which we do consider in this paper because the source KGs did not regurlarly correctly answer Step 1 queries during our experimental period (May to July, 2022). We plan to add in this web application Tables 2, 3 and 4 completed for all the profiles it displays.

Table 2 shows the number of edges (triples) and nodes in KGs on the one hand, and on the other hand the number of distinct types and terms, the number of edges (nb TT APs i.e., $|\mathbb{P}|$) and the number of maximal nodes (i.e., $|PV|$, for PV defined in Algorithm 1) appearing in their TT profile. The types extracted in first steps of TTProfiler (cf. Figure 8) are filtered in order to ignore the types of RDF, RDFS and OWL, plus some other

[39] https://ontop-vkg.org/
[40] https://www.ics.forth.gr/isl/x3ml-toolkit
[41] SPARQL endpoint: http://edan.si.edu/saam/sparql
[42] https://silknow.eu/ and SPARQL endpoint: https://data.silknow.org/sparql
[43] http://www.culturaitalia.it/opencms/index.jsp?language=en and SPARQL endpoint: http://dati.culturaitalia.it/sparql
[44] https://www.doremus.org/ and SPARQL endpoint: http://data.doremus.org/sparql
[45] https://dati.beniculturali.it/il-progetto/ and SPARQL endpoint: https://dati.beniculturali.it/sparql
[46] We use only classes of DBpedia's ontology (dbo).
[47] https://pactols.frantiq.fr
[48] https://www.getty.edu/research/tools/vocabularies/aat/

Table 2. Knowledge graphs and TT profiles

$\mathcal{A}$	statistics for $\mathcal{A}$			statistics for TT profile		
	nb. of triples	nb. of nodes	language	nb. of types & terms	nb. of AP	nb. of nodes
Epicherchell	3,945	1,438	-	31	15	13
Kition	32,714	11,280	fr	18	28	14
Iceramm	44,538	10,717	-	567	37	17
Rita	76,885	19,034	-	515	11	9
Outagr	115,462	49,719	fr	253	19	13
Chronique	557,724	183,839	fr	72	29	19
Arsol	669,099	212,179	-	93	34	17
Smithsonian	2,542,142	969,172	en	75	41	17
Silknow	4,927,819	1,843,623	en	519	477	50
Culturaitalia	41,901,551	9,951,821	en	144	271	144
Doremus	91,093,377	18,696,243	en	2,399	1,785	115
Beniculturali	755,704,024	$\geq$ 23,930,910	en	641	7,518	440
DBpedia	1,095,869,333	5,674,487	en	404	6,010	204

vocabularies depending on the KG. For instance for DBpedia we retrieve only its proper types. Although the set of basic abstract patterns is already a condensed representation of the original graph, it can be too large to be easily visualized, 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. In general it is the terms that are grouped in the same node. Notice that terms are not grouped in one node for Culturaitalia. This is because there are instantiation statements (rdf:type) with term URIs as object (those URIs being declared as skos:Concept instances), then TTProfiler considers them as types. Another noticeable fact concerns DBpedia: its terms (instances of skos:Concept) are the categories used in Wikipedia, they qualify DBpedia's entities with the dcterm:subject predicate. But as it is a hub for the LOD, DBpedia uses a lot of existing ontologies. The number appearing in the fifth column is only the number of types belonging to the DBpedia's ontology (dbo: types), and it does not comprise the number of Wikipedia categories (terms).

Function *Term* has been the subject of much experimentation and various optimizations. It can be seen as a SPARQL query corresponding to the *union of* the three patterns shown in Figure 9. More precisely, Table 3 shows, for each KG, the number of entities (or subjects) that are qualified by terms discovered, for each of the three patterns of Figure 9. Each knowledge graph uses its specific way to represent the terms, which also complicates its exploration. We notice that even the knowledge graphs that come from OpenArcheo do not all use the same pattern. For instance, Epicherchell, Iceramm, Rita, Outagr and Arsol use both skos:prefLabel and skosxl:altLabel to instantiate their terms while Kition and Chronique use only skos:prefLabel, respectively skosxl:altLabel, to label their terms. Smithsonian, Culturaitalia, Beniculturali and DBpedia use skos:Concept to instantiate their terms. Doremus and Silknow use both skos:Concept and skos:prefLabel[49]. We provide Table 3 to concretely show the important role of terms in KGs. The benefit of viewing them in profiles is demonstrated with the web API for visualizing TT profiles.

To show that, we present in Figure 10 the visualization of Chronique's TT profile. Nodes suffixed by et_al are sets of terms and colours denote namespaces, as specified at the top. A click on a node, here S19_Encounter_Event, highlights its direct neighbourhood, while putting the rest of the profile in grey. Notice that the weight of

[49]For Doremus, if we put optional the skos:prefLabel in the query then we get a "Time-Out" response, as the result is far bigger.

Table 3. Statistics of the patterns that provide the terms

$\mathcal{A}$	$nbSubjTConcept$	$nbSubjTprefLabel$	$nbSubjTxlaltLabel$
Epicherchell	0	202	182
Kition	0	1,484	0
Iceramm	0	2,109	624
Rita	0	1,586	1,247
Outagr	0	2,636	7,369
Chronique	0	0	50,863
Arsol	0	34,865	34,793
Smithsonian	60,911	0	0
Silknow	327,113	138,593	0
Culturaitalia	62,793	0	0
Doremus	4,511,304	>3,143,134	0
Beniculturali	165,362	0	0
DBpedia	1,052,223	0	0

predicates is depicted on edges. Node's content is also displayed on demand, for instance Figure 11 shows a focus on one term-node, the one that groups the terms used to qualify the use of the built archaeological artifacts described in Chronique. We invite interested readers to explore the profiles presented online, as the visualization tool well demonstrates the usefulness of TTProfiler, at least for rather small graphs. It still needs to provide ways to better navigate in the profiles when they contain too many nodes and edges. Profiles can also be explored as lists of patterns (as proposed by ABSTAT web site).

6.3 Scalability of TTProfiler

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. Its code is published in Github (footnote 9). It is devised to apply to KGs that can be queried online via a SPARQL endpoint. This requires 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. Table 4 presents the whole execution times (including time for getting SPARQL queries results from the online Endpoints) of TTProfiler for the thirteen knowledge graphs used in this paper. As we can see it, TTProfiler lasts few seconds in those that come from OpenArcheo, at most 116.528 seconds in Iceramm when data properties are recovered, and 111.963 seconds if not. In the larger ones, the execution times are significantly bigger and reach 5 hours with Beniculturali when data properties are recovered, while it lasts about 4 hours in DBpedia in the same case. We note that the data property recovery phase is time consuming with large knowledge graphs like DBpedia (2.2 hours), Beniculturali (1.39 hours), and Doremus (0.21 hours). These execution times confirm that TTProfiler is fast and can be used with large knowledge graphs without loading them locally: all TTProfiler's queries are executed online on SPARQL endpoints. We can not directly compare these execution times with related works, especially with ABSTAT, because (i) they do not run on a SPARQL Endpoint (but on various dumps of DBpedia that are loaded locally), and (ii) they do not compute exactly the same patterns. Nevertheless, even in [1] where the authors present optimized parallel implementations,

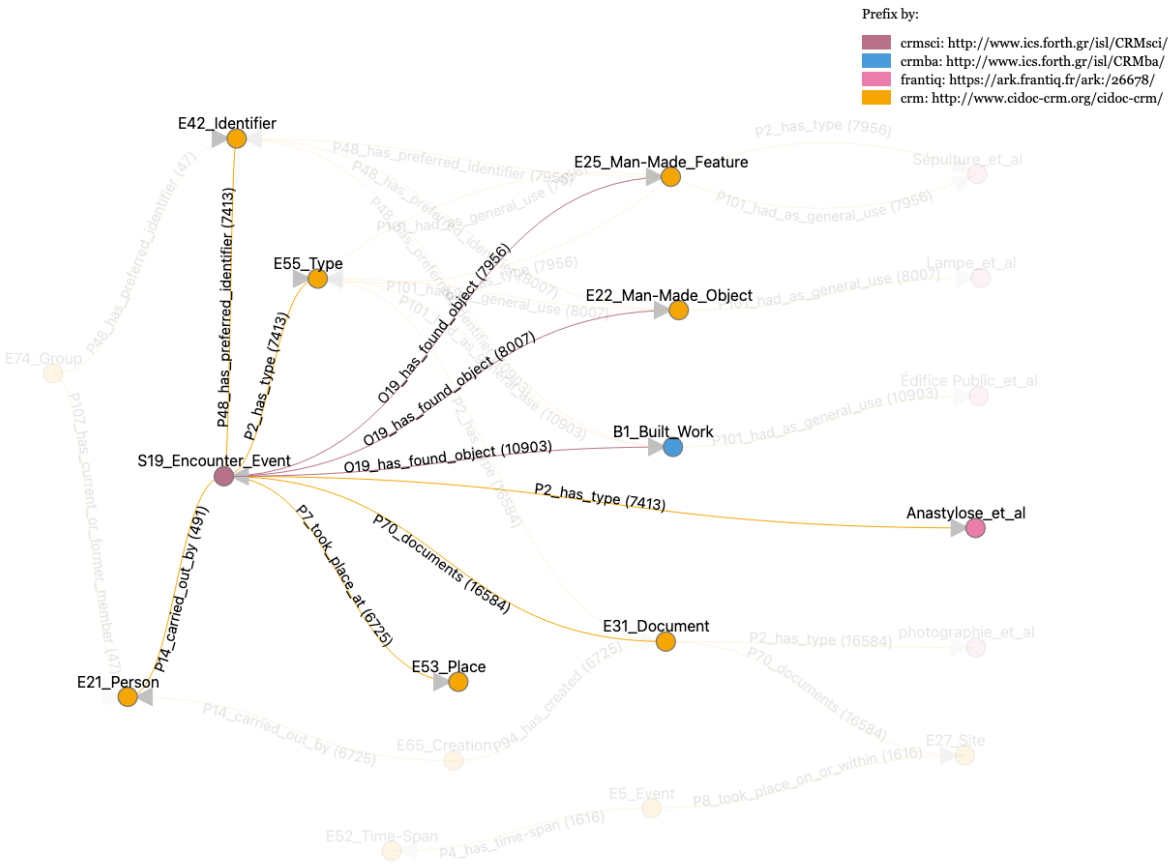

Fig. 10. Chronique's TT profile.

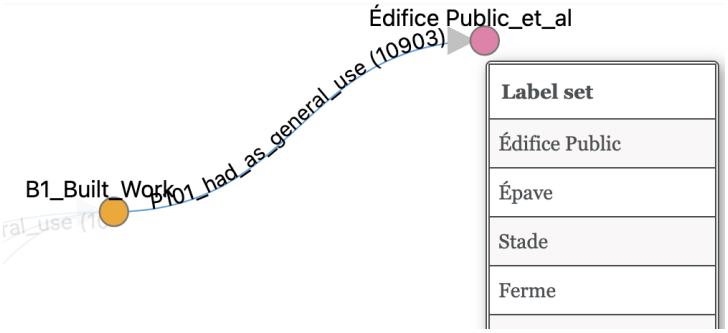

Fig. 11. Focus on a node which groups terms.

Table 4. Execution time of TTProfiler in seconds

	Epicherchell	Kition	Iceramm	Rita	Outagr	Arsol	Smithsonian
With data properties	13.741	22.917	116.528	44.307	22.742	107.557	126.857
Without data properties	10.97	18.785	111.963	40.737	18.222	95.716	90.103
Cost of data properties	2.771	4.132	4.565	3.57	4.52	11.841	36.754

	Doremus	Culturaitalia	Beniculturali	Silknow	Chronique	DBpedia
With data properties	2,717.571	333.978	18,517.262	283.717	95.724	16,385.592
Without data properties	1,966.629	288.194	13,488.834	241.521	93.433	8,453.727
Cost of data properties	750.942	45.784	5,028.428	42.196	2.291	7,931.865

ABSTAT's executions still take hours on various DBpedia dumps, therefore TTProfiler's execution appears to be reasonably efficient.

7 CONCLUSION

We presented TTProfiler, a program that extracts from a knowledge graph the predicates it uses and what they connect, which is represented by the *types* of the connected entities, and the *sets of terms* that characterize the entities in this KG. TTProfiler's implementation is findable and accessible on Github, and freely reusable. It can be easily adapted to each specific use of terminological resources in KGs. Terms play an important role in Web KGs in general, and in Cultural Heritage knowledge graphs in particular, especially for those exploiting CIDOC (because CIDOC classes and properties are abstract for the purpose of interoperability, and can be further specified by using terminologies). Ignoring the terms for these KGs when profiling them is ignoring the real topics of their contents.

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 demonstrate how interesting it is for humanists users to explore terms in TT profile visualizations. But, the usefulness of terms goes beyond humanities: categories are first class citizen in Wikipedia, and 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 it comes to give an idea about a knowledge graph's content and topics.

A TT profile can be used for supporting humans in discoverins KG's content, in order to retrieve the information they want. We have designed a public Web application to visualize the profiles generated by TTProfiler and interactively explore them, and we plan to provide also an API for applications to query the profiles. We are also studying ways of completing a TT profile with information from the ontologies, by extracting the minimal parts of ontologies concerned by the profile's types and predicates. Another need demonstrated by our experiments, is to build summaries or sampling of the profile when it is too large to be easily shown, or at least to provide zooming capabilities in order to cope with their too large number of nodes and edges.

ACKNOWLEDGMENTS

This work is supported by the ANR-18-CE38-0009 ("SESAMES"). The authors thank Zilu Yang and Lica Oka for their work on the Web visualisation tool.

REFERENCES

[1] Renzo Arturo Alva Principe, Andrea Maurino, Matteo Palmonari, Michele Ciavotta, and Blerina Spahiu. 2021. ABSTAT-HD: a scalable tool for profiling very large knowledge graphs. *The VLDB Journal* (2021).

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (Eds.). 2003. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, New York, NY, USA.

[3] Chryssoula Bekiari, George Bruseker, Martin Doerr, Christian-Emil Ore, Stephen Stead, and Athanasios Velios. 2021. Definition of the CIDOC Conceptual Reference Model. Last official version: 7.1.1. *ICOM/CIDOC Documentation Standards Group. CIDOC CRM SIG* (2021).

[4] Antonis Bikakis, Eero Hyvönen, Stéphane Jean, Béatrice Markhoff, and Alessandro Mosca. 2021. Editorial: Special Issue on Semantic Web for Cultural Heritage. *Semantic Web* 12, 2 (2021), 163–167.

[5] Stefano Borgo, Roberta Ferrario, Aldo Gangemi, Nicola Guarino, Claudio Masolo, Daniele Porello, Emilio M. Sanfilippo, and Laure Vieu. 2022. DOLCE: A descriptive ontology for linguistic and cognitive engineering. *Applied Ontology* 17, 1 (2022), 45–69.

[6] S. Cebiric, F. Goasdoue, H. Kondylakis, D. Kotzinos, I. Manolescu, G. Troullinou, and M. Zneika. 2018. Summarizing Semantic Graphs: A Survey. *The VLDB Journal* 28 (2018), 295–327.

[7] Lamine Diop, Arnaud Giacometti, Béatrice Markhoff, and Arnaud Soulet. 2021. TTProfiler: Computing Types and Terms Profiles of Assertional Knowledge Graphs. In *Semantic Web and Ontology Design for Cultural Heritage (SWODCH)*. CEUR. http://ceur-ws.org/Vol-2949/paper4.pdf

[8] F. Goasdoue, P. Guzewicz, and I. Manolescu. 2020. RDF graph summarization for first-sight structure discovery. *The VLDB Journal* 29, 5 (2020), 1191–1218.

[9] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *ACM Comput. Surv.* 54, 4 (2021), 71:1–71:37. https://doi.org/10.1145/3447772

[10] Kenza Kellou-Menouer, Nikolaos Kardoulakis, Georgia Troullinou, Zoubida Kedad, Dimitris Plexousakis, and Haridimos Kondylakis. 2021. A survey on semantic schema discovery. *The VLDB Journal* (2021), 1–36.

[11] D. Kless, L. Jansen, and S. Milton. 2016. A content-focused method for re-engineering thesauri into semantically adequate ontologies using OWL. *Semantic Web* 7, 5 (2016), 543–576.

[12] D. Kless, S. Milton, E. Kazmierczak, and J. Lindenthal. 2015. Thesaurus and ontology structure: Formal and pragmatic differences and similarities. *Journal of the Association for information science and technology* 66, 7 (2015), 1348–1366.

[13] Bernadette Farias Lóscio, Caroline Burle, and Newton Calegari. 2017. Data on the Web Best Practices (W3C Recommendation 31 January 2017): Best Practice 15. https://www.w3.org/TR/dwbp/. Accessed on 2022-28-02.

[14] Olivier Marlet, Thomas Francart, Béatrice Markhoff, and Xavier Rodier. 2019. OpenArchaeo for usable semantic interoperability. In *Open Data and Ontologies for Cultural Heritage (ODOCH)*. CEUR. http://ceur-ws.org/Vol-2375/paper1.pdf

[15] Thomas Neumann and Guido Moerkotte. 2011. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *2011 IEEE 27th International Conference on Data Engineering*. 984–994. https://doi.org/10.1109/ICDE.2011.5767868

[16] Christophe Roche and Maria Papadopoulou. 2020. Terminology and Ontology for Digital Humanities: The Case of Ancient Greek Dress. *Humanités numériques* 2 (2020). https://journals.openedition.org/revuehn/462

[17] Bertrand Russell. 2015. *On the relations of universals and particulars.* Lulu Press, Inc.

[18] Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, and Andrea Maurino. 2016. ABSTAT: Ontology-Driven Linked Data Summaries with Pattern Minimalization. In *The Semantic Web - ESWC 2016 Satellite Events, Revised Selected Papers*. Springer, 381–395.

[19] Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. 2008. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 567–580.

[20] Georgia Troullinou, Haridimos Kondylakis, Evangelia Daskalaki, and Dimitris Plexousakis. 2017. Ontology understanding without tears: The summarization approach. *Semantic Web* 8, 6 (2017), 797–815.

[21] Gang Wu, Juanzi Li, Ling Feng, and Kehong Wang. 2008. Identifying potentially important concepts and relations in an ontology. In *International Semantic Web Conference*. Springer, 33–49.

[22] Ning Zhang, Yuanyuan Tian, and Jignesh M Patel. 2010. Discovery-driven graph summarization. In *2010 IEEE 26th international conference on data engineering (ICDE 2010)*. IEEE, 880–891.

[23] Xiang Zhang, Gong Cheng, and Yuzhong Qu. 2007. Ontology summarization based on rdf sentence graph. In *Proceedings of the 16th international conference on World Wide Web*. 707–716.