Digital Healthcare Empowering Europeans
R. Cornet et al. (Eds.)

doi:10.3233/978-1-61499-512-8-924

PyMedTermino: an open-source generic API for advanced terminology services

Jean-Baptiste LAMY[a,1], Alain VENOT[a] and Catherine DUCLOS[a]
[a]*LIMICS, Université Paris 13, Sorbonne Paris Cité, Université Paris 6, INSERM UMR_S 1142, 74 rue Marcel Cachin, 93017 Bobigny, France*

Abstract. The integration of terminologies is still a challenging problem in medical informatics research and software applications, due to the high number of heterogeneous terminologies. In this paper, we present a generic API (Application Programming Interface) for a multi-terminology multilingual terminology service, and PyMedTermino, its open-source implementation in Python with 5 terminological resources (ICD10, SNOMED CT, MedDRA, CDF, VCM iconic language) and the UMLS compendium. This service has been designed for research and educational purpose. It offers various advanced functionalities rarely present in terminology services.

Keywords. Terminology as Topic, Medical Informatics/education, Unified Medical Language System

Introduction

Terminological resources play a crucial role in medical informatics research and software applications. However, the integration of terminologies is still a challenging problem, due to the high number of terminologies and their heterogeneity: monoaxial or multiaxial, single language or multilingual, pre-coordinated or post-coordinated, textual or graphical such as the VCM icons (Visualization of Concept in Medicine [1] is an iconic terminology).

Most terminologies are available from SQL databases or tabulated text files intended to be imported into a database. However, SQL requests do not allow loop nor recursion, which limits its use in training sessions. Moreover, databases are not appropriate for post-coordinated terminologies such as VCM icons: in VCM, an icon is created by combining up to 10 components (pictograms, colors, shapes,...) and the number of possible combinations is very high in theory (> 4 millions). Therefore, a generic database model is not a solution.

In this paper, we present the design of a generic API for a multi-terminology multilingual terminology service, and PyMedTermino, its open-source implementation in Python with 5 health terminologies, and the UMLS terminology compendium. It has been designed mainly for research purpose, *e.g.* batch processing of terminologies, and educational purpose, being simple enough to be used by students. It also proposes some advanced operations rarely present in terminology services.

1 Corresponding Author.

1. Materials and methods

First, a generic terminology model and a generic API were designed, with the objective of representing the elements and operations that are common to all terminologies. The model and API were based on previous research projects on VCM integration with other terminologies [2] involving four terminological resources and compendium: ICD10 (International Classification of Diseases, version 10-2010), SNOMED CT (Systematized NOmenclature of MEDicine - Clinical Terms, version 2014-01-31), UMLS (Unified Medical Language System [3], version 2012AA), and VCM. We did not try to include every operation in the generic API: terminology-specific attributes or operations were integrated into a terminology-specific API. For example, "Obtain parent concepts" is a generic operation that can be applied to any terminology, while "Obtain the dagger and star statuses of a concept" is ICD10-specific. We also tried to separate the conceptual part (natural language-independent) from the linguistic part.

In the generic model, we considered two types of operations. *Basic* operations have a terminology-dependent implementation, usually consisting in database requests or specific algorithms (for post-coordinated terminologies). An example is the "Obtain parent concepts" operation. *Derived* operations are implemented using the basic operations; they are terminology-independent and depend only on the type of terminology: monoaxial, multiaxial with or without cycles. For example the "Iterate over ancestor concepts" operation can be implemented using recursive calls to the "Obtain parent concepts" operation; however for multiaxial terminologies with cycles, the cycles must be broken to avoid infinite recursions.

Second, six terminologies were integrated: the four previously mentioned, and, later, MedDRA (Medical Dictionary for Regulatory Activities, version 17.1) and CDF

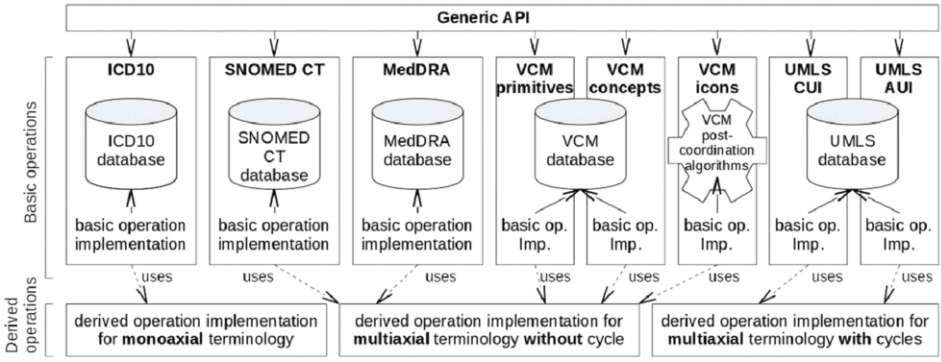

Figure 1. General architecture of the service.

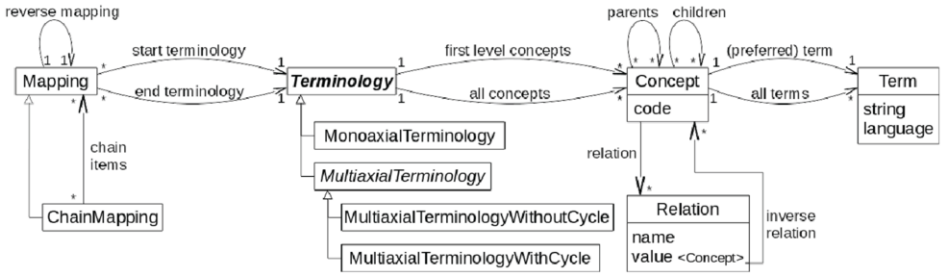

Figure 2. UML class diagram of the generic terminological model.

(CoDiFication from the Thériaque French drug databank, http://theriaque.org). The proposed model and API were implemented in Python (version 3.4). For each terminology, we implemented the basic operations and the terminology-specific operations, and we selected the appropriate implementation for derived operations (monoaxial, multiaxial with or without cycles).

UMLS was divided in two terminologies: CUI (Concept Unique Identifier: a concept in UMLS, *e.g.* "C0085580 essential hypertension") and AUI (Atom Unique Identifier: a concept in a given UMLS source terminology, *e.g.* "A0930328 Essential (primary) hypertension (ICD10)"). We also made it possible to extract a terminology from UMLS and to query it using source codes instead of UMLS AUI codes. This allows to create dynamically new terminologies such as "ICD10 from UMLS", which is considered as a different terminology from the original ICD10.

VCM was divided in three terminologies: VCM primitives (pictogram lexicon, *e.g.* "heart pictogram"), VCM concepts (concepts represented by primitives, *e.g.* "heart organ"), and VCM icons (which are a combination of up to 10 VCM primitives, *e.g.* "current color (red)" + "pathology shape (square)" + "heart pictogram"). Basic operations on VCM icons were implemented as post-coordination algorithms.

Mappings were implemented either using UMLS, or a specific database format supporting exact and partial matches (if a partial match is found for a concept, its parents are also mapped) and mapping for sets (*e.g.* {A, B} mapped to C).

The resulting implementation has been used in research projects and education.

2. Results

The general architecture of the system is shown on Figure 1. Basic and terminology-specific operations are implemented for each terminology, while derived operations implementations depend only on the structure of the terminology. Figure 2 shows the generic model and Table 1 the list of basic and derived operations. We defined advanced operations on sets of concepts, which are usually absent in existing terminology services. For example, the "lowest common ancestors" operation could be used to obtain a more general and simpler view on a long list of concepts, *e.g.* it could simplify the set of concepts {"angina", "heart failure", "valvulopathy"} into {"cardiac disorder"}. We also propose derived operations for reversing or chaining mappings (common tasks in research projects).

MedDRA and CDF have been integrated without modifying the generic model and API designed before. This argues in favor of the genericity of the model and API.

The terminology service has been implemented in Python and named PyMedTermino. It is available as open-source software (GNU LGPL licence) and can be downloaded at http://pypi.python.org/pypi/PyMedTermino. As terminologies cannot be freely redistributed, PyMedTermino does not include terminological contents. User has to obtain the appropriate rights, download terminologies and then PyMedTermino includes scripts for installing and building databases.

Figure 3 shows an example of Python script using PyMedTermino for searching SNOMED CT for concepts with "hemorrhag" in their wording but not related to the "hemorrhage" morphology (or one of its descendants). The concepts found include concepts with no hemorrhage (*e.g.* "ulcer without hemorrhage") but also concepts for which the morphology is missing (*e.g.* "viral hemorrhagic fever"). Performances are good and the script is executed in a fraction of seconds on a recent computer. Notice

On a terminology	Iterate over all concepts
	Obtain the first level concepts
	Obtain a concept from its code
	Search for a concept (free-text search)
On a concept	Obtain the code of the concept
	Obtain the preferred term in a given language
	Obtain all the terms (preferred term and synonyms) in a given language
	Obtain parent concepts
	Obtain children concepts
	Obtain the lists of available (non is-a) relations (including inverse relations)
	Obtain the values of a given relation
	Test if a concept **is a** descendant of another concept (including the other concept itself)
	Iterate over ancestor concepts, with or without doubles, including or not the concept itself
	Iterate over descendant concepts, with or without doubles, including or not the concept itself
On a set of concepts	Find all concepts in the set that are, or are not, another concept (is-a relation)
	Keep only the most generic or the most specific concepts in the set
	Computes the lowest common ancestors
	Test if set A is a *semantic* subset of set B (taking is-a relation into account)
	Perform usual set operations (union, intersection, difference, etc)
On a mapping	Map a concept from the start terminology to the end terminology
	Map a set of concepts (and remove resulting doubles, if any)
	Create the reverse mapping
	Chain the mapping to another mapping, resulting in a new mapping

Table 1. List of the basic (on white background) and derived (gray) terminological operations.

that the script could be even simpler and shorter using *set of concepts* and the "Find all concepts in the set that are another concept" operation.

PyMedTermino has been successfully used by the author in training sessions with students in master of biomedical informatics, with either a computer science background or a medical background. The example of Figure 3 was initially an exercise for the students. The generic API is interesting from an educational point of view, because it provides an exhaustive list of the various operations available on any terminology. Students can compare the various terminologies and *e.g.* see the differences between ICD10 and SNOMED CT. Terminology-based tools have a role to play in medical education [4]; for students in biomedical informatics, the generic API we propose can give them a more technical view on terminologies than the simple navigation in a terminology browser.

Script:

```
from pymedtermino.snomedct import *
for concept in SNOMEDCT.search("hemorrhag*"):
   if not concept.is_a(SNOMEDCT[404684003]): continue
   has_hemorrhage = False
   for hemorrhage in SNOMEDCT[50960005].self_and_descendants_no_double():
      if hemorrhage in concept.associated_morphology:
         has_hemorrhage = True
         break
   if not has_hemorrhage: print(concept)
```

Output:

```
SNOMEDCT[37442009] # Peptic ulcer without hemorrhage AND without perforat
ion (disorder)
SNOMEDCT[240523007] # Viral hemorrhagic fever (disorder)
... (154 concepts listed)
```

Figure 3. Example of Python script using PyMedTermino. This script searches for all concept in SNOMED CT with "hemorrhag" in one of their terms but not associated with the "hemorrhage" morphology (id 50960005) or one of its descendants. Additionally, only clinical findings (id 404684003) are considered.

3. Discussion and conclusion

Many terminology services have been proposed. HeTOP (Health multi-Terminology and Ontology Portal) [5] is a multilingual portal for browsing medical terminologies; it is linked to the CiSMEF medical search engine and it proposes webservices. Other terminology services are aimed at hospital and industrial applications. Open Terminology Services (OTS) [6] is a generic open-source API for terminology services, but seems unmaintained. LexGrid [7] is a generic framework for storing and querying various terminologies. D. Luna [8] proposed a terminology service oriented toward text search. UTS (UMLS Terminology Services) allows to query UMLS *via* webservices.

All terminology services propose similar functionalities, as noted by J. Pathak *et al.* on drug-related services [9], corresponding to the basic operations and some of the derived operations we presented (ancestors, descendants). On the contrary, the other derived operations (lowest common ancestors, *etc*) are rarely proposed. Most services also share a common architecture: a single database with a generic model that supports several terminologies. We proposed a different architecture, with a separate database and implementation for each terminology. This architecture allows to better preserve the original terminology. It also makes possible the integration of post-coordinated terminologies, which require combination algorithms in addition to a database.

The main limitation of this work is that the API is currently available only in Python. Webservices could be developed to make it usable with any programming language, however the performance would be reduced. Other limitations are the absence of support for OWL ontologies, and the quality of the UMLS mappings.

In conclusion, we presented a generic API for a terminological service aimed at research and education, and its implementation in Python. This service includes 6 heterogeneous terminological resources and compendium, and provides the usual terminological operations but also more advanced operations. The perspectives are the inclusion of additional terminologies and mappings.

Acknowledgement

This work was partly funded by the French research agency (ANR, *Agence Nationale de la Recherche*) through the SiFaDo project (ANR-11-TECS-0014), and French drug agency (ANSM, *Agence Nationale de Sécurité du Médicament et des produits de santé*) through the VIIIP project (AAP-2012-013).

References

[1] Lamy JB, Duclos C, Bar-Hen A, Ouvrard P, Venot A. An iconic language for the graphical representation of medical concepts. BMC Medical Informatics and Decision Making. 2008;8:16.
[2] Lamy JB, Tsopra R, Venot A, Duclos C. A Semi-automatic Semantic Method for Mapping SNOMED CT Concepts to VCM Icons. Stud Health Technol Inform. 2013;192:42–6.
[3] National library of medicine. Unified Medical Language System (UMLS) knowledge source; 1997.
[4] Grosjean J, Merabti T, Griffon N, Dahamna B, Darmoni SJ. Teaching medicine with a terminology/ontology portal. Stud Health Technol Inform. 2012;180:949–53.
[5] Grosjean J, Merabti T, Dahamna B, Kergourlay I, Thirion B, Soualmia LF, et al. Health multi-terminology portal: a semantic added-value for patient safety. Stud Health Technol Inform. 2011;166:129–38.
[6] Solbrig HR, Armbrust DC, Chute CG. The Open Terminology Services (OTS) project. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2003;p. 1011.
[7] Pathak J, Solbrig HR, Buntrock JD, Johnson TM, Chute CG. LexGrid: a framework for representing, storing, and querying biomedical terminologies from simple to sublime. JAMIA. 2009;16(3):305–15.
[8] Luna D, Lopez G, Otero C, Mauro A, Casanelli CT, González Bernaldo de Quirós F. Implementation of interinstitutional and transnational remote terminology services. AMIA Symposium. 2010;2010:482–6.
[9] Pathak J, Peters L, Chute CG, Bodenreider O. Comparing and evaluating terminology services application programming interfaces: RxNav, UMLSKS and LexBIG. JAMIA. 2010;17(6):714–9.