Comparing and evaluating terminology services application programming interfaces: RxNav, UMLSKS and LexBIG

Jyotishman Pathak, Lee Peters, Christopher G Chute, et al.

JAMIA 2010 17: 714-719
doi: 10.1136/jamia.2009.001149

Comparing and evaluating terminology services application programming interfaces: RxNav, UMLSKS and LexBIG

Jyotishman Pathak,[1] Lee Peters,[2] Christopher G Chute,[1] Olivier Bodenreider[2]

[1]Division of Biomedical Statistics and Informatics, Mayo Clinic, Rochester, Minnesota, USA
[2]US National Library of Medicine, Bethesda, Maryland, USA

Correspondence to
Dr Jyotishman Pathak, Division of Biomedical Statistics and Informatics, Mayo Clinic, Rochester, MN 55905, USA; pathak.jyotishman@mayo.edu

Received 23 September 2009
Accepted 29 August 2010

ABSTRACT

To facilitate the integration of terminologies into applications, various terminology services application programming interfaces (API) have been developed in the recent past. In this study, three publicly available terminology services API, RxNav, UMLSKS and LexBIG, are compared and functionally evaluated with respect to the retrieval of information from one biomedical terminology, RxNorm, to which all three services provide access. A list of queries is established covering a wide spectrum of terminology services functionalities such as finding RxNorm concepts by their name, or navigating different types of relationships. Test data were generated from the RxNorm dataset to evaluate the implementation of the functionalities in the three API. The results revealed issues with various aspects of the API implementation (eg, handling of obsolete terms by LexBIG) and documentation (eg, navigational paths used in RxNav) that were subsequently addressed by the development teams of the three API investigated. Knowledge about such discrepancies helps inform the choice of an API for a given use case.

The evolution of terminologies, across the spectrum of detailed nomenclatures and sophisticated classifications, has accelerated dramatically this decade,[1] and terminologies play a crucial role in applications including knowledge management, data integration and decision support.[2] To facilitate the integration of terminologies into applications, various terminology services application programming interfaces (API) have been developed in the recent past. In the biomedical domain, for example, such API for terminology services are a key component of the architecture of the cancer biomedical informatics grid (caBIG) developed under the auspices of the National Cancer Institute (NCI).[3]

In practice, these API are tuned to efficiently and effectively provide a host of functional characteristics ranging from retrieving concept attributes such as definitions and synonyms, to navigating relationships between concepts (eg, finding sub or super-concepts of a given concept) and accessing information combinatorially (eg, list the immediate parent concepts of all concepts that have a term that contains the word infarction). In addition, the API provide various degrees of fault resilience (to ensure high availability of service), security (to prevent unauthorized alteration and/or disruption of content) and federation (to maintain linkages among components of a large terminology, or cross-references among related terminologies).

As is true of interfaces in general, terminology service API integrated in biomedical applications have an impact on the overall quality of these applications. For example, the inability of a drug terminology API to serve the latest available data or to identify links between drug entities may cause a clinical decision support system (CDSS) relying on terminological information to make wrong inferences. For example, the following scenario illustrates the practical consequences on health care of suboptimal terminology services: A drug terminology service fails to identify the link between a brand name and its ingredients (eg, between Hamarin and allopurinol), which is used by the CDSS to identify drug–drug interactions among ingredients (eg, between allopurinol and warfarin). The CDSS, having failed to identify the proper interactions based on the information from the drug terminology service, fails to send an alert to the physician, and adverse events (eg, increased anticoagulation) occur in a patient as a consequence of the interaction between drugs (eg, metabolism of warfarin inhibited by allopurinol). Although hypothetical, this scenario illustrates the possible impact of terminology services on health care and motivates our investigation of terminology services.

In some cases, multiple terminology services API deliver overlapping capabilities and mechanisms for querying the same information, thereby making it important to evaluate the consistency and accuracy of the functionalities provided. The goal of this study is to perform a functional evaluation of three publicly available terminology services API, RxNav, UMLSKS and LexBIG, with respect to the retrieval of information from one biomedical terminology, RxNorm, to which all three services provide access.

CASE DESCRIPTION: BACKGROUND AND MATERIALS

RxNav

The RxNav is a browser for RxNorm, the repository of standard names and codes for clinical drugs developed at the US National Library of Medicine (NLM).[4] The RxNav displays links from clinical drugs, both branded and generic, to their active ingredients, drug components and related brand names. The RxNav uses a web service API (see http://www.rxnav.nlm.nih.gov/RxNormAPI.html for details) to access the RxNorm data.[5] The API provides various functionalities ranging from searching for a name in the RxNorm dataset to get the RxCUI (concept unique identifiers) to finding relationships between drug entities.

UMLSKS

The unified medical language system (UMLS), developed at the NLM, includes the metathesaurus, the semantic network and the SPECIALIST lexicon.[6 7] These resources are typically used by application programs to interpret and refine user queries, to map the user's terms to appropriate controlled vocabularies and classification schemes, to interpret natural language, and to assist in structured data creation. The UMLS knowledge source server (UMLSKS; http://www.umlsks.nlm.nih.gov/) provides access to the UMLS through both navigation and programming services.[8] The web service API was developed to support specific queries in order to reduce the total amount of information traveling between the UMLSKS and client applications, and also to provide applications with fine-grained control over the data they wish to receive.

LexBIG

The LexBIG is a community-wide project coordinated by the Mayo Clinic Division of Biomedical Statistics and Informatics for creating a vocabulary server built on a well-structured web service API capable of accessing and distributing vocabularies served by a common information model, namely, the LexGrid model.[9] This model provides the core representation for all data managed and retrieved through the LexBIG system, and is rich enough to represent vocabularies provided in numerous source formats including the UMLS rich release format (RRF), the web ontology language and open biomedical ontologies. The current implementation of LexBIG provides a robust and flexible tool set for loading, indexing and managing vocabulary content as well as Java interfaces to various functions, including lexical queries, graph representation and hierarchy traversal. Unlike RxNav and UMLSKS, LexBIG is also compliant with the HL7 common terminology services I specification.

RxNorm

The RxNorm, a standardized nomenclature for clinical drugs, is produced by the NLM.[10] It contains the names of prescription and many non-prescription formulations approved for human use (primarily in the USA). An RxNorm clinical drug name reflects the active ingredients, strengths and dose form comprising that drug. When any of these elements vary, a new RxNorm drug name is created as a separate concept. Consequently, to distinguish between such drug entities, RxNorm uses 'term types' (TTY). Furthermore, the RxNorm drug entities are related to each other by a well-defined set of named relationships (figure 1). For example, ingredient concepts (TTY=IN) are related to clinical drug component concepts (TTY=SCDC) by the relationships ingredient_of and has_ingredient. Finally, RxNorm also contains a list of identifiers from other vocabularies that appear as concept attributes (table 1).

The following materials were used in the study: (1) RxNav API 1.0 released in October 2008 and accessible via http://www.rxnav.nlm.nih.gov/RxNormAPI.html; (2) UMLSKS API 5.2 released in July 2005 and accessible via http://www.umlsks.nlm.nih.gov; (3) LexBIG API 2.3 released in October 2008 and accessible via https://www.gforge.nci.nih.gov/projects/lexevs; and (4) RxNorm November 17, 2008 full update release data that are consistent with the 2008AB version of the UMLS, and accessible via http://www.download.nlm.nih.gov/umls/kss. This dataset included 4112 ingredients, 100 dose forms, 13 923 clinical drug components, 8180 clinical drug forms, 18 228 clinical drugs, 10 029 brand names, 14 154 branded drug components, 11 643 branded drug forms, 14 891 branded drugs, 288 branded packs and 224 generic packs. Furthermore, the dataset had over 500 000 relationships between these RxNorm entities.

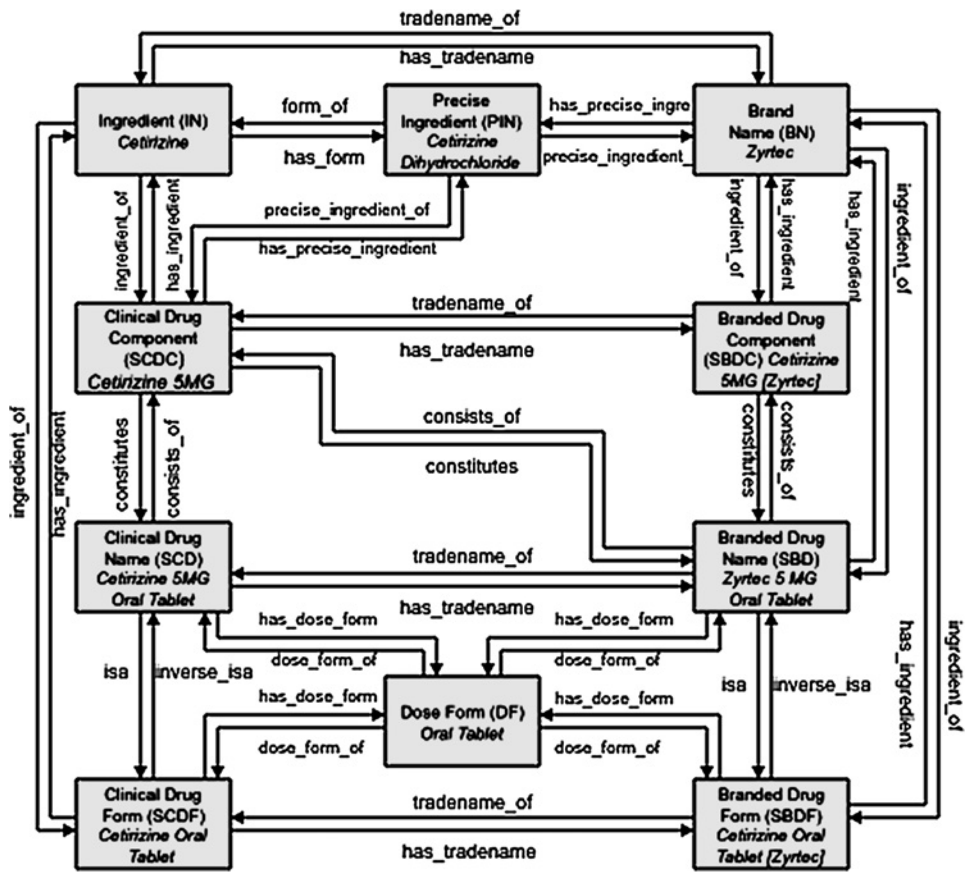
tradename_of
has_tradename
Ingredient (IN) Cetirizine
form_of
has_form
Precise Ingredient (PIN) Cetirizine Dihydrochloride
has_precise_ingro
precise_ingredient_
Brand Name (BN) Zyrtec
ingredient_of
has_ingredient
precise_ingredient_of
has_precise_ingredient
Clinical Drug Component (SCDC) Cetirizine 5MG
Branded Drug Component (SBDC) Cetirizine 5MG [Zyrtec]
consists_of
constitutes
Clinical Drug Name (SCD) Cetirizine 5MG Oral Tablet
Branded Drug Name (SBD) Zyrtec 5 MG Oral Tablet
has_dose_form
dose_form_of
isa
inverse_isa
Dose Form (DF) Oral Tablet
Clinical Drug Form (SCDF) Cetirizine Oral Tablet
Branded Drug Form (SBDF) Cetirizine Oral Tablet [Zyrtec]

Figure 1 Relationship between RxNorm drug entities (adapted from Zeng *et al*[4]).

Table 1 RxNorm vocabulary identifiers

RxNorm idType	Identifier name
AMPID	Alchemy marked product identifier
GCN	Generic code number
GFC	Generic formula code
GPPC	Generic product packing code
GS	Gold standard alchemy identifier
LISTING_SEQ_NO	FDA identification number
MMSL_CODE	Multum identifier
NDC	National drug code
SNOMEDCT	SNOMEDCT identifier
SPL	Standard product label
UMLSCUI	UMLS concept unique identifier
VUID	VHA unique identifier

METHOD OF IMPLEMENTATION

This experiment was developed as a functional evaluation, that is, a qualitative exploration of the major terminology services functionalities for drug names and codes, such as finding drug concepts by their names or codes, or navigating different types of relationships among drug entities. Therefore, we did not record any API scalability, performance or usability information, but focused on assessing qualitative and functional discrepancies among API.

Preliminary test

In order to eliminate major issues in the process of comparing the three API, we started by performing a preliminary test, ie, by running a small number of queries and comparing the results. Careful inspection of these preliminary results revealed a few unexpected issues with the RxNav API, in particular with the restriction of multiple search results to the first candidate, and with the processing of brand and generic packs. As we intended to use the RxNav API as our reference in this study, these errors were fixed in RxNav before running the experiment. Analogously, the preliminary test also revealed an issue with string searches, also fixed before running the experiment, because it impacted the UMLS interface. A detailed account of these issues is provided in the discussion section.

Establishing reference queries and test data

We first established a list of queries (box 1) that cover a wide spectrum of terminology services functionalities, using the current implementation of the RxNav API as a guide. The RxNav API allows users to search RxNorm entities by name (Q1) and by identifier (Q2), including the proprietary identifiers from source vocabularies. Moreover, the RxNav API provides access to the various properties of an RxNorm entity, including the national drug codes (NDC) used for inventory purposes (Q3), RxNorm name (Q4), proprietary names and codes from source vocabularies (Q5). Finally, the RxNav API can be used to retrieve various types of relations among RxNorm entities, by type of relationship (Q6), by type of target entity (Q7) and all relations (Q8). Based on these API calls, test data were generated from the RxNorm dataset to evaluate the implementation of the functionalities. For each function, we randomly selected from RxNorm a list of test values (ie, a list of drug names and codes to be tested). A distinct set of 100 test values was selected for each simple function (eg, returning all NDC codes for a given drug). For complex functions, we selected larger sets of test values in order to reflect all possible variants of the query (eg, combinations of drug codes and types of relationships for the function used for exploring related drug entities for various types of relationships).

Box 1 Query functionalities tested in RxNav, UMLSKS and LexBIG (function names within the square brackets refer to the RxNav implementation)

Q1: Search for a name in the RxNorm dataset and return the RxCUI of concepts that have that name as an RxNorm term or as a synonym of an RxNorm term. [findRxCuiByString(searchString)]
Q2: Search for an identifier from another vocabulary and return the RxCUI of concepts that have an RxNorm term as a synonym or have that identifier as an attribute. [findRxCuiById(idType,id)]
Q3: Get the NDC for an RxNorm concept. [getNDCs(rxcui)]
Q4: Get the RxNorm concept properties. [getRxConceptProperties (rxcui)]
Q5: Get the concept information associated with the concept for the specified sources. The user must have a valid UMLS license and be able to access the UMLSKS authority service to obtain proxy tickets to use this function. [getProprietaryInformation (rxcui,sourceList,proxyTicket)]
Q6: Get the related RxNorm identifiers of an RxNorm concept specified by a relational attribute list. [getRelatedByRelationship (rxcui,relaList)]
Q7: Get the related RxNorm identifiers of an RxNorm concept specified by one or more types. [getRelatedByType(rxcui,typeList)]
Q8: Get all the related RxNorm concepts for a given RxNorm identifier. [getAllRelatedInfo(rxcui)]

Reference paths in the RxNorm graph

In order to query for relationships between various RxNorm drug entities, a reference list of paths among categories of entities in RxNorm was developed (see the RxNav API documentation for details). For example, as illustrated in figure 1, given the brand name Zyrtec (RxCUI=58930), the reference path to the ingredient cetirizine (RxCUI=20610) traverses the direct path between BN and IN via the relationship tradename_of. On the other hand, among all possible paths between the two drug entities, the reference path to the clinical drug cetirizine 5 mg oral tablet (RxCUI=315025) from Zyrtec is the indirect path between BN and SCD (through SBD) via the relationships ingredient_of and tradename_of.

Evaluation

In order to facilitate the comparison of the result sets from the three API, an XML Schema was established that was loosely based on the simple object access protocol envelope of the RxNav web service API (refer to the WSDL schema from: http://www.rxnav.nlm.nih.gov/RxNormDBService.wsdl). The output results generated by UMLSKS and LexBIG were evaluated against the result set from RxNav. We selected RxNav as the reference, because it was designed specifically for the dataset under investigation, RxNorm. However, all discrepancies were reviewed manually, without assuming that RxNav would necessarily return the correct answer. In other words, RxNav was used as a reference only in order to simplify the evaluation. The RxNav results were not used as the gold standard, but reviewed as critically as the results from UMLSKS and LexBIG.

This study is a qualitative not quantitative evaluation. We believe there is no acceptable proportion of discrepancies among API, and our goal is to identify reasons for discrepancies, not merely quantifying them. Special attention was given to distinguishing between differences in the underlying datasets and differences in the behavior of the various API for a given query.

RESULTS AND OBSERVATIONS

Table 2 summarizes the results for UMLSKS and LexBIG API compared with the RxNav API. The first column indicates the type of query evaluated (as listed in box 1) along with the number of queries executed (in the test data) in the second column. The third and fourth columns refer to the number of queries that differed in the results from UMLSKS and LexBIG API, respectively, compared with the results from the RxNav API for the test data. In all the cases, many differences were observed between the results returned by the individual API: we distinguish between differences in the API functionalities and differences and issues in the dataset and how it was initially loaded for querying.

Querying for drug entities by name (Q1)

For Q1, 19 differences were observed between the RxNav and UMLSKS result sets, of which 17 were caused by slight differences between the two datasets used for querying RxNav and UMLSKS. In particular, even though the RxNorm November 17, 2008 full update release data were aligned with the 2008AB version of the UMLS (used for querying the UMLSKS), some RxNorm concepts were missing from the UMLS release. The other two differences were not associated with the dataset alignment issues: the first difference occurred in searching for 'psyllium husk'. UMLSKS returned two RxCUI, 104129 ('psyllium husk') and 8928 ('psyllium'), although the second RxCUI was not found by RxNav. Investigating further, we realized that UMLSKS found 8298 because 'psyllium husk' is a synonym from the NCI thesaurus for 'psyllium', and the RxNorm dataset does not contain terms from the NCI thesaurus. The second difference occurred because of different exact match rules between the two API—the search for 'Senna Lax' yielded RxCUI 219861 ('Senna Lax') and 219864 ('Sennalax') in UMLSKS, while RxNav only found 219861 ('Senna Lax'). On the other hand, for LexBIG, out of six differences, three were due to search strings not found by the LexBIG (and found by RxNav), two were due to LexBIG returning obsolete RxNorm concepts, and finally, LexBIG returned one RxCUI without an RxNorm term.

Querying for drug entities by code (Q2)

For Q2, there were 124 differences between UMLSKS and RxNav result sets, of which 100 were due to the lack of existing capability in UMLSKS to search by idType=NDC. (Unlike other codes, NDC are stored in the attribute table of the UMLS metathesaurus, which is not amenable to search through the API). The remaining 24 differences were a result of imperfect alignment between the RxNorm and UMLS datasets as elucidated above. For LexBIG, 66 differences were observed with the RxNav result sets. In particular, for idType=GCN, LexBIG returned 29 RxCUI, which belonged to obsolete RxNorm data. In addition, one of the returned RxCUI had ET as its term type, which is invalid. Similar observations were made for idType=LISTING_SEQ_NO, in which one of the returned RxCUI pointed to obsolete data and another RxCUI did not have an RxNorm term (this was also true for idType=SNOMEDCT). Interestingly, for idType=NDC, LexBIG not only returned all the expected RxCUI, but also found 33 NDC identifiers associated with more than one RxCUI. The RxNav API, on the other hand, returned only one RxCUI per identifier, a behavior that can be attributed to the RxNav interface selecting one identifier for visualization purposes. Furthermore, for this evaluation, the results from idType=MMSL_Code were excluded because RxNav did not find any matches, due to the fact that the identifiers were not in the format required by the API.

Querying NDC for a drug (Q3)

For Q3, four differences were observed between the UMLSKS and RxNav results that were due to imperfect alignment between the RxNorm and UMLS datasets. On the other hand, LexBIG and RxNav results had no differences.

Querying the properties of a drug concept (Q4)

For Q4, five differences were observed between the UMLSKS and RxNav results, and all were a result of imperfect alignment between the RxNorm and UMLS datasets. On the other hand, there were no differences between the LexBIG and RxNav results.

Querying the proprietary information about a drug concept (Q5)

For Q5, four differences were observed between the UMLSKS and RxNav results, and all were a result of imperfect alignment between the RxNorm and UMLS datasets. On the other hand, LexBIG could not return results for any of the queries becausee such information is not captured by the RxNorm (RRF) loader for LexBIG.

Querying related drugs by relationship (Q6)

For Q6, 58 differences were observed between the UMLSKS and RxNav results, and all were a result of imperfect alignment between the RxNorm and UMLS datasets. Although, for LexBIG, 66 differences were observed, and all of them were due to the result of LexBIG returning obsolete RxNorm concepts.

Querying related drugs by type of drug entity (Q7)

For Q7, 40 differences were observed between the UMLSKS and RxNav results, of which 38 were due to the imperfect alignment between the RxNorm and UMLS datasets. The other two differences were observed when UMLSKS retrieved the desired

Table 2 UMLSKS and LexBIG query result comparison with the RxNav API

		Query results with UMLSKS differences		Query results with LexBIG differences	
Query type	Total no of queries executed	API related	Non-API related	API related	Non-API related
Q1 (find by name)	820	2	17	3	3
Q2 (find by code)	1100	100	24	33	32
Q3 (get NDC codes)	100	0	4	0	0
Q4 (get concept properties)	102	0	5	0	0
Q5 (get proprietary information)	100	0	4	0	100
Q6 (get related drugs by relationship)	1060	0	58	0	66
Q7 (get related drugs by type of drug)	820	2	40	19	61
Q8 (get all related drugs)	100	1	17	16	19

results, but RxNav failed due to issues in processing branded and generic packs (BPCK and GPCK), for example, Tri-Pak, a branded pack of three oral tablets of azithromycin. For LexBIG, we observed 80 differences with the RxNav results, and the bulk of which (61) were due to LexBIG returning obsolete data. In addition, 16 differences were observed when the target type was IN, and LexBIG results were missing the precise ingredients (PIN), and one difference occurred (RxCUI=236216, endTTY=SCD) in which the LexBIG results did not return the SCD associated with the PIN. Similar to UMLSKS, the other two differences observed were due to issues in RxNav processing of branded and generic packs.

Querying all related drugs (Q8)

For Q8, 18 differences were observed between the UMLSKS and RxNav results, of which 17 were a result of imperfectly aligned RxNorm datasets. One difference occurred (RxCUI=494944) in which UMLSKS results did not find a dose form concept (DF). On the other hand, 37 results were different in LexBIG compared with RxNav, of which 19 differences were due to LexBIG returning obsolete RxNorm concepts. For the remainder, 16 differences were observed when the target type was IN, and LexBIG results were missing the precise ingredients (PIN), one difference occurred (RxCUI=2625) in which LexBIG did not return the clinical drugs associated with the PIN, and finally, another difference occurred (RxCUI=494944) when LexBIG did not find a DF concept.

DISCUSSION

Practical implications

As illustrated in our evaluation, leaving aside minor nuances, all the three API were functionally similar in terms of information retrieval, and differences in result sets were primarily due to issues in dataset alignment and content loading. However, the investigation of these discrepancies was beneficial and prompted changes in each of the three API. Knowledge about such discrepancies helps inform the choice of an API for a given use case.

Implications for LexBIG

In particular, for LexBIG, major differences in results were due to returning obsolete RxNorm concepts for the queries performed. The information about obsolete concepts, although present in the RxNorm dataset, was not captured by the RRF loader in LexBIG. Similarly, all the concept information associated with a concept for the specified sources was not captured by the RRF loader in a consistent way. For instance, the information about the mapping between an RxNorm concept (eg, RxCui=161 ('acetaminophen')) and a concept in another vocabulary (eg, id=5005 in MMSL) was missing in many cases, and as a consequence, LexBIG could not return results for query Q5 (ie, for retrieving proprietary names and codes from source vocabularies in RxNorm). We created an entry for this issue in the LexBIG bugtracker that was subsequently addressed by the development team. However, at the same time, the LexBIG API was highly performant and provided various convenience methods uniformly to query its underlying common information LexGrid model.

Implications for UMLSKS

For UMLSKS, the differences observed in result sets were mainly due to one reason: imperfect alignment of the 2008AB release of UMLS with the RxNorm November 17, 2008 full update release. While addressing this issue is beyond the scope of this work, we realized that typically the RxNorm dataset is submitted to the UMLS maintainers a few weeks before the UMLS scheduled release date. Consequently, by the time the UMLS metathesaurus is made publicly available, the RxNorm dataset would have evolved due to the addition of new drugs or the elimination of obsolete ones, thereby causing the RxNorm dataset to have included new data (without UMLS CUI information) as well as eliminated old data. The UMLSKS API also did not explore all the features of the RxNorm dataset. For example, it was not possible to search for RxCUI using NDC identifiers. In addition, during a preliminary analysis, it was discovered that UMLSKS incorrectly returned drug concepts searched by name, when the search string had more than 30 characters. For example, when executing query Q1 (finding drug entities by name) for the string 'benztropine injectable solution', UMLSKS returned RxCUI 371036 ('benztropine injectable solution') and 92198 ('benztropine injectable solution (Cogentin)'), of which the latter is incorrect. After notifying the UMLSKS maintainers about this issue, a problem in the string-matching algorithm was discovered, and subsequently fixed before analyzing our test data. One of the search features in UMLSKS that could benefit RxNav is the removal of special characters from the search string for exact matches. For example, searching for 'senna lax' yielded RxCUI 219861 ('senna lax') and 219864 ('sennalax') in UMLSKS, but RxNav only retrieved 219861. It is worth noting that LexBIG already implements such a feature.

Implications for RxNav

Although RxNav was used as a reference for evaluating discrepancies among API, its results were not assumed as a gold standard, but reviewed critically during the preliminary test, leading to the identification of issues in RxNav. In addition to issues with exact match string searching, we discovered problems involving the traversal of the RxNorm graph for drug packs (TTY=GPCK and BPCK) in which the reference path was not used. For example, when executing query Q6 (querying related drugs by type of drug relationship) with RxCUI=750119 ('Tirosint 0.013 56 day pack') and TTY=SCDC, RxNav returned no results. The correct result is the concept 'thyroxine 0.013 mg' (RxCUI=728558), which was returned by both UMLSKS and LexBIG. Furthermore, when querying drug entities by code (Q2), RxNav returned only one RxCUI per identifier (such as NDC), although the dataset contained more than one RxCUI in some cases. Finally, we observed that documentation for few functionalities implemented by RxNav was sparse and required significant enhancements. In particular, the documentation about the RxNorm graph traversal as well as relationship mappings between RxNorm entities required improvement. All these issues were brought to the RxNav development team's attention during our investigation and have been subsequently addressed.

Implications for the choice of an API

Similar to differences observed among SNOMED-CT browsers in an earlier study,[11] we believe this study and exercise is of significance since without such a systematic analysis, it is nontrivial for the users to identify API-level implementation bugs. In particular, by identifying reasons for discrepancies among API, this study was useful to the developers of the API as it gave them the opportunity for identifying and fixing several API implementation errors. The remaining discrepancies come from differences in the datasets and in the inherent capabilities of the API. Knowledge about these discrepancies, along with technical and functional specifications, can help inform the choice of an API for specific use cases, that is, align the requirements of an application with the capabilities of a given API.

Both RxNav and LexBIG load and import the RxNorm dataset directly, and can therefore be pointed to the latest version, whereas the UMLSKS API can only serve the version of RxNorm integrated in a given edition of the UMLS. Given the differences in frequency of update between RxNorm (monthly) and UMLS (biannually) and possible residual discrepancies for a given version of RxNorm (as illustrated in this study), the UMLSKS API is not the right choice for querying an up-to-date RxNorm dataset. Moreover, the UMLSKS API also does not allow drugs to be queried by NDC. However, because it can query the entire UMLS, the UMLSKS API is useful in cases when drug entities need to be linked to other entities (eg, diseases), whereas the RxNav API is limited to querying the RxNorm dataset. Analogously, LexBIG has the ability to load and serve terminologies developed using non-RRF languages, such as web ontology language and open biomedical ontologies—a feature that is not available in both UMLSKS and RxNav API.

Although evaluating the quality of the documentation was beyond the scope of this study, we found that adequate instructions and code samples were provided for the three API investigated to support application developers. Because the three API provide web services for accessing RxNorm data, programming language requirements are not a factor in the choice of an API, because most modern programming languages provide support for web services. One small difference among the three API lies in the protocols used by the API. Whereas all three API support simple object access protocol-based web services (simple object access protocol), RxNav and LexBIG also support the REST protocol (representational state transfer).

All three API provide some form of access control. Therefore, this element should not be determinant in the choice of an API. Due to licensing requirements, an active UMLS metathesaurus license code is required to access the complete UMLS release. The UMLSKS API has robust authentication mechanisms for granting 'proxy tickets' to access the UMLS data. The RxNav API requires one such proxy ticket for using the function getProprietaryInformation, through which proprietary names and codes are exposed. In LexBIG, a particular terminology has to be designated as 'active' by the administrator to make it queryable. Therefore, terminologies whose access is restricted can be made 'inactive' for access control purposes.

Finally, LexBIG API provides a much broader set of functionalities compared with UMLSKS and RxNav. Examples include defining and querying value sets, terminology mappings, terminology version management and so on. UMLSKS and RxNav, on the other hand, are limited to querying for concepts, their attributes and relationships in the terminology of interest (UMLS metathesaurus, semantic network, and SPECIALIST lexicon for UMLSKS, and RxNorm for RxNav).

Limitations and future work

The selection of queries in this study does not reflect any specific use case (eg, medication reconciliation) or even the frequency of queries sent to the RxNav service. Rather, we elected to perform a systematic investigation of the queries available through the RxNav API. Specific evaluation schemes could be designed to investigate the degree to which atomic queries can be combined in order to support specific use cases. For example, medication reconciliation (using RxNorm as an interlingua) would require that drug codes be mapped from the source vocabulary to RxNorm (RxCUI) and then that RxCUI be expressed into the codes of the target drug vocabulary.

Several factors were purposely omitted from our evaluation. Beyond the ability of API to retrieve basic information from a terminology, the specific requirements of an application should also be taken into account in the choice of an API, for example, in terms of required functionalities and communication protocols. Another aspect of our investigation that requires further evaluation is analyzing performance and reliability (eg, running performance benchmarks and analyzing fault resilience). Finally, evaluating the quality of the documentation was beyond the scope of our investigation.

The study only evaluated the retrieval of information from one biomedical terminology, RxNorm. In the future, we plan to expand our investigation by incorporating more terminology sources, although arguably, API such as RxNav, developed specifically for a particular terminology, will not be applicable. Furthermore, we intend to include additional publicly available terminology services API such as Apelon DTS[12] in our study.

CONCLUSIONS

In this study, we experimented with three publicly available terminology services API to query a clinical drug terminology, RxNorm. The main finding of this study is that the three API investigated are essentially functionally similar. However, the systematic investigation of discrepancies among the three API highlighted various issues in each API that have since been addressed. The other benefit of this study is that knowledge about such discrepancies helps inform the choice of an API for a given use case. Finally, our investigation, the first of its kind, contributed to provide a methodological model in comparing and evaluating terminology services API developed by different organizations.

Acknowledgments The authors would like to acknowledge the inputs and critiques from the 2009 AMIA Annual Symposium reviewers.

Funding This research is supported in part by NHGRI grant U01HG04599 (eMERGE) and the Intramural Research Program at NLM.

Provenance and peer review Not commissioned; externally peer reviewed.

REFERENCES

1. **Cimino J,** Zhu X. *The practical impact of ontologies on biomedical informatics*. IMIA Yearbook 2006: Assessing information—technologies for health. International Medical Informatics Association, 2006:124—35.
2. **Bodenreider O.** Biomedical ontologies in action: role in knowledge management, data integration and decision support. *IMIA Yearbook 2008: Access to Health Information*. International Medical Informatics Association, 2008:67—79.
3. **Komatsoulis G,** Warzel DB, Hartel FW, *et al*. caCORE version 3: implementation of a model driven, service-oriented architecture for semantic interoperability. *J Biomed Inf* 2008;**41**:106—23.
4. **Zeng K,** Bodenrider O, Kilbourne J, *et al*. RxNav: towards an integrated view on drug information. *Medinfo* 2007:P386.
5. **Peters L,** Bodenreider O. Using the RxNorm web services API for quality assurance purposes. *AMIA Ann Symp Proc* 2008:591—5.
6. **Lindberg DA,** Humphreys BL, McCray AT. The unified medical language system. *Methods Inf Med* 1993;**32**:281—91.
7. **Bodenreider O.** The unified medical language system (UMLS): integrating biomedical terminology. *Nucl Acids Res* 2004;**32**(database issue):D267—70.
8. **Thorn KE,** Bangalore A, Browne A. Plug-and-play UMLS knowledge source server using web services and portlets. *AMIA Ann Symp Proc* 2006:1121.
9. **Pathak J,** Solbrig HR, Buntrock JD, *et al*. LexGrid: a framework for representing, storing, and querying biomedical terminologies from simple to sublime. *J Am Med Inform Assoc* 2009;**16**:305—15.
10. **Liu S,** Ma W, Moore R, *et al*. RxNorm: prescription for electronic drug information exchange. *IT Pro* 2005;**7**:17—23.
11. **Rogers J,** Bodenrieder O. *SNOMED CT: browsing the browsers. In 3rd International Conference on Knowledge Representation in Medicine (KR-MED 2008)*. Phoenix, AZ, 2008:30—6.
12. Apelon Distributed Terminology Server (DTS). http://www.apelon-dts.sourceforge.net (accessed 7 May 2010).