A Robust, Self-Training Classifier for Medication Strings, with Quantitative and Semantic Confidence Metrics

Mark A. MILLER[a,1], Hayden FREEDMAN[a], Christian J. STOECKERT, JR [a,b]
[a]*Institute for Biomedical Informatics, Perelman School of Medicine, University of Pennsylvania, 3700 Hamilton Walk, Philadelphia, PA, 19104, United States*
[b]*Department of Genetics, Perelman School of Medicine, University of Pennsylvania, 415 Curie Boulevard, Philadelphia, PA, 19104, United States*

Abstract. The TURBO Medication Mapper (TMM) identifies terms from RxNorm that best represent a list of medication strings, like one would find in a Clinical Data Warehouse (CDW). TMM has several differentiating characteristics, compared to other tools: the machine learning component does not require a human-curated gold standard for training; normalizations are applied to source-specific language in the strings (instead of just excluding them); the confidence of each mapping is represented as a relationship to the absolute truth, along with a 0.0-1.0 score; the results, along with supporting knowledge, are saved into an RDF graph and a Solr document database is generated. Queries for drug classes like "statins" are based on OBO foundry ontologies like the Drug Ontology (DrOn) and ChEBI, and they return more results than multiple SQL search strategies over the CDW, with few false positives. TMM is available for download from GitHub.

Keywords. EHR, medications, classifier

1. Introduction

Electronic Health Records (EHR), Clinical Data Warehouses and related datasets play an invaluable role in the cohort-building phase of translational research, i.e. determining which patients to include. Data columns or fields that use consistent, unambiguous values (or codes) for patient characteristics are immediately actionable. Consider an exclusion criterion like "patients must have no recorded body mass index of 30 or higher".

Medication orders, another common component of cohort definitions, are the subject of this paper. In the ideal system, patient identifiers would be linked to encoded medications with rich, unambiguous semantics like "Vytorin tablets contain simvastatin and ezetimibe". That would make the medication records as directly actionable as discrete values like BMI. Free-text mentions of medications in clinical notes fall at the other end of the usability spectrum, as they require NLP techniques like grammatical parsing and named entity recognition.

This paper specifically addresses a task whose complexity lies in the middle of that usability spectrum: encoding the meaning of medication strings, like one would find in

[1] Corresponding Author. markampa@pennmedicine.upenn.edu.

an EHR's or CDW's medication name column. Generally, these lists are strongly enriched for ingredient names, brand names, routes and or strengths, so there is great potential for processing all of those data efficiently and in bulk. If a software tool could encode the medication strings, according to a vocabulary interoperable with multiple semantic or ontological models, then additional knowledge of the drug's therapeutic intent, side effects, etc. could be exposed.

In our CDW, the `MEDICATION` table is a heterogeneous mix of signals like those listed above, along with noise, like devices, medical supplies, nutritional products, lab orders, patient notes, etc. While the majority of heterogeneity and noise appears to come from legacy data in the CDW, it can also be observed in data loaded from the current production EHR, which presumably encourages the use of pick lists. Additionally, the drug orders in our CDW come from health care encounters with > 100 types, including emergency, inpatient, outpatient, office visit, phone visit, etc.

In cases like this, heterogeneity and noise can lead to the use of regular expressions and value sets, both of which can be difficult to maintain or properly attribute. Consider searching for patients with orders for HMG-CoA reductase inhibitor drugs from the same structural series as mevastatin, or "people taking statins" [1] in the words of most clinicians and patients alike. An SQL query based on the presence of the substring "statin" [2] requires negation for seven unrelated drugs like the antifungal nystatin, as well as the inclusion of nine brand names, since some orders do not mention an active ingredient. In our CDW, brand names and active ingredients can both be found in a `FULL_NAME` column. Ingredients are also present in the `GENERIC_NAME` column, which should also be searched for "statin", plus the negations above. This calls for a roughly 950 character long SQL query that will require local maintenance with the introduction of new drugs. When run against our CDW, the described query retrieves 610 medications. Manual inspection suggests that the false positive rate is essentially zero.

The CDW also provides `PHARMACY_SUBCLASS` annotations for a subset of the medications. Unfortunately, an SQL search that takes a more semantic approach by requiring a `PHARMACY_SUBCLASS` value of 'HMG-CoA Reductase Inhibitors' only returns 207 drugs. Table 1 shows the proportion of drugs annotated with the top five `PHARMACY_SUBCLASS`es for both an unqualified 'statin' substring search and for the search with several negations.

Table 1. `PHARMACY_SUBCLASS` annotations returned by two different SQL searches for statin drugs. Only annotations that appeared in 1% of the with-negation results are shown. Many drugs are not annotated with any `PHARMACY_SUBCLASS`.

PHARMACY_SUBCLASS	%, 'statin' substring query	%, query with negation
HMG CoA Reductase Inhibitors	38.1	70.2
Cardiovascular Agents Misc. - Combinations	4.8	11.2
Antihyperlipidemics - Combinations	4.4	8.7
Antidiabetic Combinations	2.7	5.0
Nutritional Supplements	2.0	2.5

In the PennTURBO initiative [3], we perform RDF instantiation of patients, the processes they participate in, the data items that are generated by those processes, and more, all according to the principles of the OBO Foundry. We also import OBO ontologies like DrOn and ChEBI to model drug product composition, branding, clinically relevant structural classes, and biochemical roles. The role of this new TURBO

Medication Mapper (TMM) is to rapidly generate an easily updatable RDF graph of which drugs, according to RxNorm, were ordered for which patients. The classes and predicates used by TMM come from the TMM ontology, which follows RDFS and SKOS semantics more than OBO semantics. However, the whole landscape of patient knowledge can be tied back together in OBO style with one subsequent SPARQL statement. [Not shown] From the perspective of someone looking for patients with orders for a "statin", TMM provides solutions for all of the steps in the following strategy:

1. Determining which prescribable products (in a brand or ingredient sense), are mentioned by the medications modelled in the EHR/CDW
2. Determining which drug ingredients belong to the statin class, in terms of their structure or biochemical activity
3. Determining which modelled medications contain the statin ingredients
4. Determining who received orders for those medications

Step 1 is largely lexical, or based on the character composition of different strings. Those comparisons could be made based on rigid rules, like the SQL queries above, but TMM uses a more flexible machine learning approach. Steps 2 thorough 4 are more semantic in nature, i.e. concerned with entities that exist in the universe (of medications, in this case) and the ways in which they are related.

There are numerous tools [4, 5, 6, 7] for addressing these problems, frequently as part of a broader system for encoding clinical records including notes. For example, numerous tools can address step 1 by mapping textual medication orders to Concept Unique Identifiers (CUIs) from the Unified Medical Language System (UMLS) [8,9] or from UMLS' medication-focused RxNorm subset [10]. Once step 1 has been completed, steps 2 and 3 can be pursued within RxNorm, or within the UMLS ecosystem, with a moderate degree of semantic precision.

Unfortunately, the lexical technologies in many of these tools require either the software developer or the user to maintain a dictionary derived from one or more authoritative knowledgebases like RxNorm. There may also be a statistical model for aligning the medication orders with the dictionary and setting cutoff scores or confidence values. These maintenance requirements may cause the software tool to lag behind the approval and availability of new drugs. And even if these tools map a peculiar medication string to an RxNorm CUI with moderate confidence (because a perfect match isn't present in RxNorm), there may be no explanation for why the score is only moderate. How can we model the fact that a real patient took a real drug that just doesn't happen to be modeled by RxNorm? Perhaps we can describe the way their drug is related to an RxNorm entity, using RxNorm's own language.

TMM mappings are built with lexical and semantic approaches; they report confidence from quantitative and semantic standpoints; and they enable users to access the modeled data through lexical and semantic channels. The lexical matching is performed against all the labels of all RxNorm term types, but the semantic portion enables searching against realism-based knowledge in OBO-foundry ontologies like ChEBI ([11], for active ingredients, the clinically relevant structural classes they belong to, and the drug roles they bear) and DrOn ([12], for the relationship between orderable products and ingredients).

TMM requires no annotated, source-specific "gold-standard" training data, although it can accept and generally benefits from normalization rules. By normalization, we mean replacing source specific language in the EHR ("po tabs") with RxNorm language bearing the same meaning ("oral tablet"). The generation of these normalization rules is currently largely manual, and the rules themselves are stored in a CSV configuration file. Our normalization can also drop several of the tokens from "ALTEPLASE (TPA) IV BOLUS FOR ACUTE ISCHEMIC STROKE" as they are noise for our purposes: they don't say anything about the drug product itself.

TMM explicitly anticipates that, for many of the medication strings, it won't be possible to map to an RxNorm with perfect fidelity. In most cases, each medication string is initially mapped to multiple RxNorm terms. For each RxNorm mapping, the probability that the RxNorm term perfectly captures the detail of the medication is reported. Additionally, TMM reports the probability that an RxNorm mapping is off by one well-characterized relationship in the RxNorm graph. Imagine a patient obtained 565 mg atorvastatin tablets while travelling outside of the US. When the string "565 mg atorvastatin tablets" is processed by TMM, an exact match will not be found in RxNorm, which only models drugs approved for sales in the US. Nonetheless, TMM will still map that string to multiple RxNorm terms, including "atorvastatin Oral Tablet", RxCUI 370621. One could represent confidence about the mappings with tuples like {("identical", 0.01) ("has ingredient", 0.01) ("has dose form", 0.95) ("more distant" 0.01)…} TMM also includes methods for electing the best of the multiple mappings.

Where possible, TMM uses existing tools and strategies instead of reinventing the wheel. Our choices for components were largely influenced by commitments to fast, robust performance and especially to maintainability. Medication-specific search tools like RxNav [13] (or even general purpose search tools like Solr [14]) can return high-quality scored and ranked lists of possible encodings for a medication string. Unfortunately, these systems do not guarantee that the results include a perfect mapping, or if one is present, that it appears as the top-ranked match. Therefore, TMM calculates additional lexical and relevance measures for each match and then uses machine learning to predict not just "which is best" but "what makes it a good match".

In remainder of this paper, we will describe

- Patterns in a CDW that could make medication mapping challenging, with examples from an otherwise high-quality CDW commonly used at the University of Pennsylvania
- TMM methods for building a semantic graph that includes knowledge about patients, medication orders, and medications in general
- Methods for searching the TMM graph from text-relevance and semantic perspectives
- A comparison between the searches of the CDW versus TMM

1. Materials and Methods

TMM applies the previously described normalization to medication strings and then submits them to the RxNav search engine. RxNav returns matches, with scores based on the Jaccard index between the tokens in the input and those in in the matches. Then TMM adds features like multiple string similarity measures between the inputs and matches (lexical), `rxcui.count` (a relevance metric, specifically the number of other medication strings that share the same RxNorm match), the semantic type of the matching RxNorm term (TTY, like "ingredient", "brand name", "dose form"), and the upstream source from which RxNorm obtained the match's label or synonyms (SAB). These features are used as inputs into a random forest (RF) classifier that simply learns how to interpret the features as predictors of the semantic relations between searches and matches ("has ingredient", "brand name of", etc.). TMM also imports additional knowledgebases and builds a Solr index, so that users can query the data with lexical and semantic approaches, akin to the construction of the overall knowledgebase. Prototypical queries retrieve or count patients whose medication orders follow patterns like: contains an ingredient like "atorvastatin"; branded, with a name like "Lipitor"; contains an ingredient that belongs to a structural class like "statins" or bears a biochemical role like "HMG-CoA reductase inhibitor"

1.1 CDW Background

We have been granted access to an Oracle-based CDW that integrates records from Penn's current and legacy EHRs. All medications known to the CDW are modelled in a MEDICATION table, which has columns such as a Primary key, a foreign key to the `ORDER` table, `FULL_NAME, GENERIC_NAME,` and a sparsely filled `RXNORM.` By joining the `ORDER` table's foreign key to the `ENCOUNTER` table (which contains a patient identifier column), it is possible to count the number of patients for whom any given `MEDICATION` was ordered. In order to improve TMM's performance and to keep protected health information (PHI) out of the final product, we don't process any `MEDICATION` that was ordered for fewer than two distinct patients.

As of early May, 2020, the `MEDICATION` table had 948k records. The most commonly ordered `MEDICATION` is "ACETAMINOPHEN 325 MG PO TABS", which has been ordered for 308,946 unique patients. However, most medications have been ordered for a very low number of patients: 132,748 medications ordered for only one patient, and 700,117 medications tied to zero patients. Among the 115,573 `MEDICATIONs` ordered for 2+ patients, 28.4% are already RxNorm-annotated. Of those, 2,127 are no longer present in the 2019AB release of RxNorm. Therefore, TMM uses an RxNorm-vs-RxNorm self-training approach, instead of using the source RxNorm terms as a gold standard for training the RF.

Implementation

TMM is implemented with several scripts in the R language (tested under versions 3.5 though 3.6). The current version has been executed on Ubuntu Linux 18 LTS and Apple Mac OS Catalina systems. TMM is unlikely to work on any system with less than 32 GB RAM. See supplemental materials for required R libraries and additional details. [2]

TMM also requires Solr (tested with 7.3.1), a local installation of the RxNav-in-a-box Docker environment, and a Docker engine compatible with docker-compose, with one small configuration change. [2]. We are currently using Solr 7.3.1 and Docker Desktop 2.3.0.2 on our Apple systems and Docker CE 19.03.8 on our Ubuntu systems. In a similar vein, TMM submits requests for inter-ontology term mappings, in bulk, to the NCBO BioPortal. Specifically, we obtained mappings between RxNorm, DrOn and ChEBI (TMM's core public datasets) with a local installation of OntoPortal 2.5, which returns results quickly and minimizes demands on public resources. Downloading

RxNav-in-a-box and the OntoPortal virtualization files require UMLS and BioPortal accounts and the submission of access requests.

TMM is designed to use an RDF triplestore database to hold its results, along with RDF serializations of the core public datasets. Specifically, TMM has been designed to use OntoText's GraphDB. The process of querying TMM or submitting updates via some other triplestore is completely possible with minimal modification, but the process of loading the core public datasets and a RDF representation of the mapping process could be significantly different with other triplestores. We have used GraphDB versions 8.x through 10.x. GraphDB requires Java 1.8. The free version of GraphDB is suitable as long as only one user will be submitting queries or updates at a time.

Internally, TMM generates a tabular representation of source medications, search results and classifications. Because TMM uses a triplestore as durable storage, a conversion to RDF is required. Several libraries are available for converting tables to RDF in R, but we have not been satisfied with their performance when it comes to generating triples from millions of rows by 50 columns. Therefore TMM exports the tables in a form that is compatible with the Java-based ROBOT [15], which performs the RDF generation and is therefore another requirement. All columns are asserted as annotation properties; ROBOT's more advanced OWL expression features are not used.

TMM is available for download, with documentation, from GitHub [16]. It does have several dependencies, but can realistically be put into use by informaticians familiar with technologies like GitHub, R, Solr, and semantic web approaches in general. The software isn't tightly coupled to particular versions of the public knowledgebases, so users can update them without having to wait for a new TMM release.

User provided inputs

Once the dependencies have been installed, users should create a table with the medications from their system in TMM's format (Table 2). We recommend writing a custom R script that queries their EHR or CDW and saves the results as `source_medications.Rdata` in R's binary format, in order to avoid problems with character encoding or with special characters, like quotation marks, carriage returns and other non-printing characters.

Table 2. TMM's format for `source_medications.Rdata`, the serialization of the medications to be classified

Column	Necessity	Notes
`MEDICATION_ID`	required	Unique alphanumeric identifier for the medication string
`FULL_NAME`	required	The medication string to be mapped
`GENERIC_NAME`	Optional input	A generic "equivalent" for the `FULL_NAME`. Will also be considered as a mapping input if the `FULL_NAME` doesn't align well with any strings in RxNorm
`RXNORM`	Optional (QC)	RxNorm RxCUIs, if/when available from the source, for comparison with TMM's RxCUI predictions
`MEDICATION_COUNT`	Optional (filtering, QC)	An indication of how frequently orders for this `FULL_NAME` appear in the source system

For the previously described string normalization, TMM also requires a `med_name_normalization.csv` file with five columns, three of which are parsed (Table 3). Users can store notes in the additional `mask` and `Notes` columns. A template with examples is provided [2]. `med_name_normalization.csv` could even be left entirely blank besides the five headers. However, we have found that poor mappings can frequently be attributed to the presence of terms in the `FULL_NAME` that rarely if ever appear in RxNorm strings. For example, "ACUTE ISCHEMIC STROKE" appears in many of our CDW's `MEDICATION FULL_NAME`s, but it doesn't appear in any RxNorm strings, because it isn't a characteristic of any medication. Many other papers about EHR-wide medication mapping describe this same problem and mention the use of a manually created exclusion list, although they do not provide executable code [17, 18].

We agree that some tokens need to be excluded and that automated generation of an exclusion list is an unsolved problem. Fortunately, the extent of this problem is dramatically decreased in our case by the two patient minimum. Beyond that, `med_name_normalization.csv` can be used to declare both "drop any appearance of 'ACUTE ISCHEMIC STROKE'" and "replace 'po tabs' with 'oral tablet'".

Table 3. Columns from `med_name_normalization.csv` that are parsed by TMM

Column	Necessity	Notes
`pattern`	required	A pattern that is common in the EHR or CDW medication strings, but rare in, or absent from RxNorm strings
`replacement`	optional	A substring that is (relatively) common in RxNorm's strings and has the same meaning as `pattern`. If blank, appearances of pattern will be removed without replacement
`confidence`	required	Only rows marked "high" will be used as normalization rules

Algorithm

`rxnav_med_mapping_proximity_training_no_tuning.R` is used to train TMM's RF. Like most of these R scripts, it begins by importing `rxnav_med_mapping_setup.R`, and parses `rxnav_med_mapping.yaml`. A template for this file is provided but users must enter site-specific details. This script begins by retrieving a random number of medication labels from RxNav, which are lowercased, uniquified, normalized, and submitted to RxNav's approximateTerm REST endpoint. RxNav reports identifiers for the matches, but not strings. String distances are a valuable feature for training the RF, so we obtain the match strings, the upstream source (SAB) and the string type (TTY) with a subsequent SQL query against RxNav in a box. `rxnav_med_mapping_proximity_training_no_tuning.R`. TMM also calculates the number of characters and space-delimited words in query and match strings, along with the total number of times each RxCUI appears in the search results.

The matches' RxNorm identifiers stand in place of medications that exist in our universe and can share relations with one another, like "has brand name" and "is ingredient of". Because we want the TMM RF to predict relations between inputs and search results, another SQL query is submitted to RxNav in order to retrieve all of the relations between all of the entities known to RxNorm. When the input and a search result are semantically identical, TMM asserts the non-RxNorm relation “identical”. When no direct relationship is observed between the search term and a match, the non-RxNorm relation, “more distant” is asserted. The RF is trained after splitting the feature rows and known relationships into training and validation subsets. Then it is saved, and assessed with a typical multi-label confusion table. We also provide an example script for interactive tuning of the RF.

Up to this point, the “unknowns” and the “gold standards" relations have both come from RxNav, so the “unknowns” haven’t required any normalization. Next, `rxnav_med_mapping_proximity_classifier.R` is used to normalize the `FULL_NAME`s from `source_medications.Rdata` according to the patterns in `med_name_normalization.csv.` After this, they are to the approximate match REST endpoint and additional features are generated as above. The RF is applied to those features in order to predict the relations between the unknowns and the search results. After applying the RF, each search result will have one confidence score for each of the relations defined by RxNorm, along with “identical” and “more distant”. Even though one relation will have the highest probability for each search result, the number of classified search results for each input could be zero to 50. A low pass, destructive filter is applied at this point: if there are any “identical” results, only the one(s) with the highest probability are retained. If there are no “identical” classifications, then the results classified with a single RxNorm relation are evaluated, and regardless of the relation type, the one with the highest score is retained. Finally, in the cases where all of the search results for a given input are classified as “more distant”, the one(s) with the highest score for anything other than “more distant” are retained. After this filtering, the R script exports its internal tabular data structures, which are converted into RDF with ROBOT.

NCBO BioPortal term mappings in the core public ontologies should be retrieved and converted to RDF with `serialize_bioportal_mappings.R`. These mappings support our SPARQL queries in cases where there are gaps in the axioms provided by RxNorm, DrOn and CHEBI.

`rxnav_med_mapping_load_materialize_etc.R` loads each of the previously mentioned RDF files into an isolated named graph in GraphDB, which scopes queries and improves their performance. This script implements other performance improvements by executing several SPARQL updates that transitively materialize subclass relationships and convert complex OWL restrictions into shortcut relations like “has ingredient” and “bears role”, also into isolated named graphs.

Results

RF characteristics

The scripts described above generate 18 features that could be used to train a random forest [2]. Five of those features were discarded due to correlation of > 0.95 with at least one retained feature, or due to low importance with respect to the mean Gini purity score. Our RF was trained over 73k search result rows, with 351 trees per forest and 9 simultaneous features per tree. The resulting accuracy (as determined with a validation set) was 0.938. The optimal tree count and simultaneous features are found on a stable plateau that was previously discovered by training RFs over a grid covering 100 to 580 trees and 5 to 12 features

With the exception of "more distant", all of the classifications, including "identical" have > 0.95 specificity (Table 4). "Identical" has reasonable sensitivity (0.822), but the sensitivities of the other classes defined by an RxNorm relation only have an average sensitivity of 0.35. Conversely, "more distant" has excellent sensitivity (0.973) but lower specificity (0.755). In fact, almost all of the misclassification can be attributed to over-inclusion in "more distant". Finally, this script addresses the fact that a medication string could still have multiple mappings, even after the low-pass filter, by electing the RxNorm term whose sum of "identical" scores is highest. Note that the prevalence of the different RF classifications are very uneven, they are consistent with the training data. 'contained in' relationships, which describe the make-up of a drug package, are just rare in the universe as modelled by RxNorm.

Table 4. TMM's classification performance for "identical", "more distant", and the best and worst scoring relations. These statistics, including prevalence, refer to the results before low pass filtering or election.

Train = Training input. Res = Random Forest Results. Prev = Prevalence.

Relationship Class	Train. Prev.	Res. Prev.	Sensitivity	Specificity	Precision	Recall	F1	Balanced Accuracy
has form	0.003	0.003	0.869	1.000	0.945	0.869	0.905	0.934
identical	0.224	0.237	0.822	0.974	0.907	0.822	0.862	0.898
more distant	0.683	0.673	0.973	0.755	0.891	0.973	0.930	0.864
contained in	0.000	0.001	0.013	1.000	0.250	0.013	0.024	0.506

TMM search results, with "statin" examples

A Solr query was run in < 1 second to determine that the "statins" structural class is best modelled as <http://purl.obolibrary.org/obo/CHEBI_87631>, and a SPARQL query was run in < 2 seconds to find 1,002 medications in our CDW with a path back to <http://purl.obolibrary.org/obo/CHEBI_87631>, compared to the 610 statins found by an SQL statement based on substring searching with negation [2].

The CDW contains 115,752 "common" medications strings, meaning that they were ordered for at least two patients. TMM mapped 108,995 of those to some RxNorm term. Table 5 shows details about the accuracy of the TMM semantic search over the random forest results, plus three SQL queries described over the course of this paper. Figure 1 is provided as background regarding the number of patients with orders for any given medication, as the impact of those counts is addressed along with the accuracy.

Table 5. Accuracy of three different SQL searches for statin drugs vs TMM semantic search over Random Forest results. T = true, F= false, P = Positive, N = Negative, Sn = sensitivity, Sp = specificity.

Search Strategy	Hits	TP	FP	FN	TN	Sn	Sp	F1
TMM: contains ingredient from statin structural class	1002	988	14	31	114719	0.970	1.000	0.984
CDW SQL: substring present in lowercased (FULL_NAME or GENERIC_NAME), with negations	610	607	3	412	114730	0.596	1.000	0.747
CDW SQL: substring only: lcase(FULL_NAME) like "%statin%"	883	554	329	465	114404	0.544	0.997	0.704
CDW SQL: PHARMACY_SUBCLASS= 'HMG CoA Reductase Inhibitors'	146	146	0	873	114733	0.143	1.000	0.251

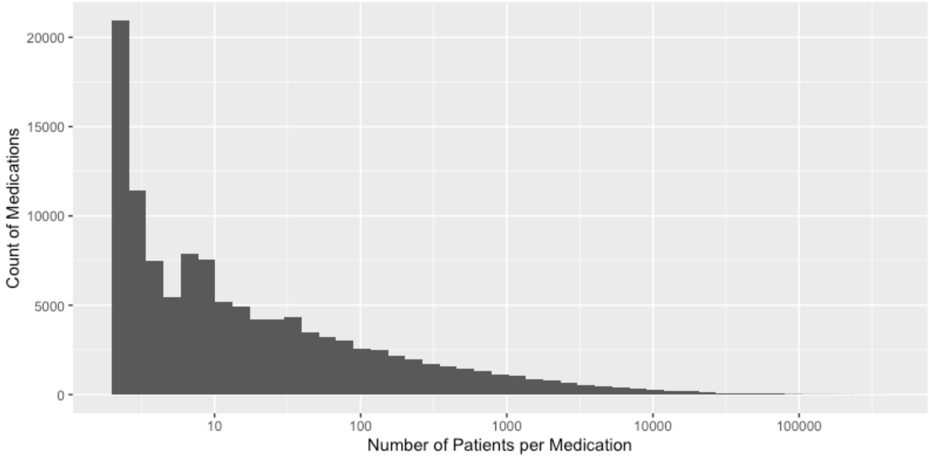

Figure 1. Overview of the number of patients per drug order. X axis starts at 2.

The intersection and disjoints between TMM's semantic statin search and the most semantic (although least productive) SQL search for "'HMG CoA Reductase Inhibitors' were manually evaluated. 136 of the common medications were identified both as HMG CoA Reductase Inhibitors in the CDW and as statins by TMM. 'ATORVASTATIN CALCIUM 40 MG PO TABS' was ordered for the greatest number of patients: 63,334.

Ten common medications were annotated as HMG CoA Reductase Inhibitors in the CDW but were not classified as statins by TMM. For 9 of the 10 medications missed by TMM, it was RxNav that did not find any matches. In all of these cases, manual review showed that the medication was not modelled in the current RxNorm. The most common medication in this category was 'BAYCOL 0.4 MG OR TABS', which was ordered for 141 patients. Relative to the CDW, TMM's only outright false negative for statins was 'LOVASTATIN 10 MG OR TB24', with 95 orders. The fact that "TB24" means "24 hour tablet" was absent from the normalization file, so that could have degraded the mapping.

It appears that 13 of the TMM only identifications were false positives. '.Amlodipine Tablet' was the worst offender with 5727 orders. 'niacin ER -' was ordered for '1013' unique patients. We believe that these 'statin' misclassifications might be attributed to the random forest trainer's exposure to medications in which those compounds appear along with statins in combination drugs. Examples of other misclassifications included '.Sodium Hypochlorite Soln0.13% (1/4 Stg)' with 7 orders, 'HYALURONATE SODIUM (EMOLLIENT) 0.1 % EX LOTN' with 3 orders, and 'Please have a Basic Metabolic Panel and Magnesium and Phosphorus level drawn on Thursday (1/12), Monday (1/16), Thursday (1/19), and Monday (1/23).' with 3 orders.

866 medications were identified as statins by TMM but were not annotated as HMG CoA Reductase Inhibitors in the CDW. The most common medication in this category was 'atorvastatin -' with 40,937 orders. Similar patterns were observed for 'pravastatin -', 'rosuvastatin -' and 'rosuvastatin -'. Those are naturally found by the best SQL search (‘statin” substring, with negation). Of the 866 TMM-only statins, 811 had been imported into the CDW from a legacy system and 55 were loaded from the current production EHR. Overall, 59737 orders were from the legacy system and 56015 orders were from the production system.

Discussion

We envision several ongoing improvements to TMM. The number of “more distant” mappings could probably be decreased with paths between medication strings and RxNorm terms over two RxNorm relations, like “ezetimibe 10 MG / Simvastatin 10” `has_tradename` “Vytorin Pill” `has_ingredient` “Vytorin”. However, there are 26 RxNorm relations, so the cartesian product might result in an impractical number of paths and decreased performance. RxNorm has monthly full releases in the UMLS’s Rich Release Format (RRF), but we have been getting an RDF version from the BioPortal, which only has biannual releases. We are currently developing our own RxNorm RRF → RDF converter.

We have described methods for filtering and electing mappings, so that only the highest scoring matches are retained. In rare cases, multiple mappings with the same score still persist. The election process could possibly be improved by looking for the most central node (from a graph perspective) or a lowest common subsumer (from a hierarchical perspective.) We could also check that the RxNorm term is even compatible as the object of the predicted relationship. For example, ingredients can’t “have ingredients”.

Some of our clinical collaborators have observed false positive, false negative or generally under-specific role and class mappings in the public ontologies we import. Fortunately, the maintainers of the public knowledgebases we have mentioned are receptive to feedback leading to updates. We would still like to better characterize therapeutic indications (mechanism of action, physiological effect, etc.) present in the ATC and NDF-RT subsets of UMLS, although their semantics don’t have same rigor as OBO foundry ontologies. We also plan to add a systemic method for rapidly capturing our collaborators’ alternative classifications, naturally using named graphs.

Conclusions

When applied to our CDW, TMM finds more “statin” medications than SQL queries over the medication strings, the sparse pharmaceutical classifications or the sparse legacy RxNorm classifications. When we have done deep dives on other medication classes like analgesics (opioid or nan-narcotic), we have found the same result [data not shown]. The relationships between these classes, ingredients and products are backed with sound semantics.

We thank O. Bodenreider, M.R. Boland, L. Davidson, J. Graybeal, W Hogan, K. Long, J. Overton and P. Wolf for valuable conversations and assistance with the technical resources that they manage. Thanks to H. Williams for random forest tuning and software usability advice.

References

[1] Stossel TP. The discovery of statins. Cell. 2008 Sep 19;134(6):903-5.

[2] https://github.com/PennTURBO/medication-knowledgegraph-pipeline/blob/master/turbo_medmapping_supplement.md

[3] Stoeckert CJ, Birtwell D, Freedman H, Miller M, Williams H. Transforming and Unifying Research with Biomedical Ontologies. In Proceedings of the 9th International Conference on Biological Ontology (ICBO 2018)

[4] Jiang M, Wu Y, Shah A, Priyanka P, Denny JC, Xu H. Extracting and standardizing medication information in clinical text–the MedEx-UIMA system. AMIA Summits on Translational Science Proceedings. 2014;2014:37.

[5] Soysal E, Wang J, Jiang M, Wu Y, Pakhomov S, Liu H, Xu H. CLAMP–a toolkit for efficiently building customized clinical natural language processing pipelines. Journal of the American Medical Informatics Association. 2018 Mar 1;25(3):331-6.

[6] Savova GK, Masanz JJ, Ogren PV, Zheng J, Sohn S, Kipper-Schuler KC, Chute CG. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. Journal of the American Medical Informatics Association. 2010 Sep 1;17(5):507-13.

[7] Aronson AR, Lang FM. An overview of MetaMap: historical perspective and recent advances. Journal of the American Medical Informatics Association. 2010 May 1;17(3):229-36.

[8] Lindberg DA, Humphreys BL, McCray AT. The unified medical language system. Yearbook of Medical Informatics. 1993;2(01):41-51.

[9] Bodenreider O. The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research. 2004 Jan 1;32(suppl_1):D267-70.

[10] Nelson SJ, Zeng K, Kilbourne J, Powell T, Moore R. Normalized names for clinical drugs: RxNorm at 6 years. Journal of the American Medical Informatics Association. 2011 Jul 1;18(4):441-8.

[11] Hastings J, Owen G, Dekker A, Ennis M, Kale N, Muthukrishnan V, Turner S, Swainston N, Mendes P, Steinbeck C. ChEBI in 2016: Improved services and an expanding collection of metabolites. Nucleic acids research. 2016 Jan 4;44(D1):D1214-9.

[12] Hanna J, Joseph E, Brochhausen M, Hogan WR. Building a drug ontology based on RxNorm and other sources. Journal of biomedical semantics. 2013 Dec 1;4(1):44.

[13] Zeng K, Bodenreider O, Kilbourne J, Nelson SJ. RxNav: a web service for standard drug information. In AMIA Annual Symposium Proceedings 2006 (Vol. 2006, p. 1156). American Medical Informatics Association.

[14] https://lucene.apache.org/solr/

[15] Jackson RC, Balhoff JP, Douglass E, Harris NL, Mungall CJ, Overton JA. ROBOT: A Tool for Automating Ontology Workflows. BMC bioinformatics. 2019 Dec 1;20(1):407.

[16] https://github.com/PennTURBO/medication-knowledgegraph-pipeline

[17] Levin MA, Krol M, Doshi AM, Reich DL. Extraction and mapping of drug names from free text to a standardized nomenclature. In AMIA Annual Symposium Proceedings 2007 (Vol. 2007, p. 438). American Medical Informatics Association.

[18] Davidson L, Boland, MR MA. Comparative Analysis and Evaluation of State-of-the-Art Medication Mapping Tools to Transform a Local Medication Terminology to RxNorm. In AMIA Summit 2020.