Design and Evaluation of an Algorithmic Parser for Free-Text Prescription Data in an Ambulatory Electronic Health Record

by

Bimal R. Desai, M.D.

A CAPSTONE PROJECT

Presented to the Department of Medical Informatics & Clinical Epidemiology
and the Oregon Health & Science University School of Medicine
in partial fulfillment of
the requirements for the degree of
Master of Biomedical Informatics
May 2008

School of Medicine

Oregon Health & Science University

Certificate of Approval

This is to certify that the Master's Capstone Project of

Bimal R. Desai, M.D.

"Design and Evaluation of an Algorithmic Parser for
Free-Text Prescription Data in an
Ambulatory Electronic Health Record"

Has been approved

Capstone Advisor -Dean F. Sittig, Ph.D.

TABLE OF CONTENTS

Acknowledgements

My time at Oregon Health & Science University would not have been possible without the support of The Children's Hospital of Philadelphia and the career mentorship provided by my Division Chief, Louis Bell. Since 2004, Lou has acted as a tireless advocate on my behalf, giving me the career flexibility I needed to complete my coursework, and helping me to chart my path in the field of clinical informatics.

Many of the skills I needed to complete this project I learned from the dedicated faculty in the Department of Medical Informatics & Clinical Epidemiology at Oregon Health & Science University. I hope to model my own professional career in Clinical Informatics after the example they have set.

I owe my sincerest gratitude to Dean Sittig, my Capstone advisor. I first met Dean as a student in his class three years ago and was immediately impressed by his deep understanding of both the technical and human domains of Clinical Informatics. This project would not have been possible without his steady guidance, enthusiastic collaboration, and unbiased feedback.

Finally, I would like to thank my wife, Naomi, whose assistance in analyzing the data was invaluable. More importantly, I thank her for her love, patience, and unfaltering support of my decision to pursue a degree in informatics. I dedicate this Capstone project to her.

Abstract

Background
Prescription data in computerized provider order entry systems contain a wealth of information about medication utilization, appropriateness of therapy, and provider prescribing habits. Unfortunately, the *signatura* or "*sig*" portion of the prescription which specifies the dose, route, frequency, duration, and other instructions is often stored as a single free-text field, making detailed analysis of large numbers of prescriptions impractical. For informaticists and clinical researchers, an accurate, automated method of analyzing free-text prescriptions would be immensely valuable.

Objective
The objective of this project was to develop and validate a software algorithm that can identify standard components of any free-text ambulatory *sig*, such as the dose, route, frequency, interval, and duration.

Design & Measurement
I wrote and validated a prescription parsing algorithm named RxParse. The parser was first tested and debugged against a subset of ambulatory prescriptions obtained from the EpicCare ambulatory health record at The Children's Hospital of Philadelphia. RxParse was then validated against a separate set of 1000 prescriptions. The performance of the algorithm was compared to that of two human reviewers. For each of eight *sig* components, I calculated the sensitivity, specificity, positive-predictive value, and negative predictive value of the parser. Finally, RxParse was run against datasets ranging from 10,000 to 350,000 *sigs* in order to measure the throughput of the parsing algorithm as measured by total elapsed time and scripts parsed per second.

Results
RxParse had an overall recall of 87% and precision of over 99%. Performance for individual dose components was more variable. Elements such as the command, dose, alternate dose, and dose interval had high precision and recall (97% or greater), while other elements such as the route had lower recall and lower negative predictive value due to limitations in the algorithm logic. Because of its regular expression-based pattern-matching algorithm, the parser showed exceptional precision across all components. The parser is capable of sustained throughput of nearly 2,000 prescriptions per second across a broad range of dataset sizes.

Conclusion
RxParse shows promise and supports the use of regular expressions for parsing of semi-structured and densely abbreviated medical text. The parser's speed makes it an ideal tool for data mining of vast prescription databases. Future work will include improvements to the parser's route detection algorithm and cross-validation using an adult prescription dataset.

I. Introduction

I.A. Background and Significance

Computerized provider order entry (CPOE) systems allow for, among other features, the digital entry and storage of prescription data. With time, these data stores can become an expansive and valuable source of information about the prescribing habits of a healthcare organization. For example, at The Children's Hospital of Philadelphia (CHOP), there are currently over two million prescriptions stored in the EpicCare ambulatory electronic health record (Epic Systems Corporation, Verona, WI). Not surprisingly, a number of health services researchers at CHOP have ongoing or proposed projects that rely on data contained in the ambulatory electronic health record and, more specifically, CPOE data. These projects include analysis of off-label drug use in pediatrics, appropriateness of therapeutic escalation for chronic conditions such as asthma, and evaluation of guideline-based dosing of medications with variable doses.

There is, however, a significant drawback to EpicCare's representation and storage of these electronic prescriptions. Currently, the *signatura* or "*sig*" portion of the prescription that represents the dosing and administration directions is stored as a single free-text field in a large database table. A typical *sig* might state "take 2 tabs PO BID x 5 days" where "tabs" is an abbreviation for "tablets", "PO" the Latin abbreviation for "by mouth", and "BID" the Latin abbreviation for "two times per day". Of note, a single *sig* can contain many discrete elements of

information about the prescription, such as the dose, the route, the unit of measure, and the interval/frequency (Table 1).

Table 1: A single *signatura* can contain many discrete elements.

Sig	Dose	Route	Frequency	Duration
take 2 tabs po bid x 5 days	2 tablets	po = oral	2 times per day	5 days

Any research question that pertains to a *sig* component requires that the researcher first parse and extract the *sig* components for further analysis. At CHOP, there is currently no automated way to parse *sigs* into their various components, instead requiring manual review to overcome the limitations of free-text data entry. This limitation functionally restricts the kinds of questions researchers can ask related to CPOE-derived data. For example, a researcher interested in the appropriate use of high-dose amoxicillin (an oral antibiotic) in the treatment of acute otitis media (middle ear infection) would have to manually review each amoxicillin prescription for the target population, since the dosing information is contained in the free-text *sig*. For any standardized evaluation of electronic prescription data – whether for quality improvement efforts, for clinical research, or for administrative purposes – *having a tool that allows users to quickly and accurately parse prescription data in an automated fashion would be immensely valuable*.

I.B. Project Objectives

Recognizing the potential value to clinical researchers and informaticists, I decided to design and evaluate a free-text prescription parser implemented in Perl, an open source scripting language (The Perl Foundation, Grand Ledge, MI). From the outset, the project had a set of key objectives:

1. **Given a text *sig*, parse the dose and frequency/interval discretely to allow subsequent calculations.** For example, given a *sig* and the patient's weight, the researcher should be able to calculate the weight-based dose of a medication. Similarly, given a *sig* and both the start and stop dates, the researcher should be able to calculate the patient's cumulative dose. At a minimum, the parsed data structure should include elements that support these derived calculations.
2. **Where available, the parser would make use of existing data standards and vocabularies.** By extension, where a CHOP-specific or EpicCare-specific vocabulary was in use, that vocabulary would be preferred.
3. **The design of the parser should allow incorporation of the tool into other research applications at CHOP.**
4. **The parser should be validated against an actual set of prescription data to assure researchers of its reliability and performance.**

I.C. Electronic Representation of Prescription Data

The choice of data model to represent parsed prescription data is non-trivial, as it impacts the generalizability of the parser to other datasets and CPOE systems. For that reason, I first conducted a review of existing efforts to represent prescription data electronically.

One of the hallmarks of electronic health records is their support of data retrieval and categorization through structured data entry. In turn, this highlights the importance of medical data standards for codification of clinical data.[1,2] Efforts to define standards for representation of prescription data have varied in focus and applicability. I reviewed the following data models for this project: NCPDP SCRIPT,[3] NHS Dose Syntax,[4] and OpenEHR Medication Archetype.[5]

NCPDP SCRIPT: This widely used representation of prescription data was developed by The National Council for Prescription Drug Programs (NCPDP), a not-for-profit organization accredited by the American National Standards Institute (ANSI).[6] NCPDP's SCRIPT standard is now in its 10th version and is implemented in the RxHub National Patient Health Information Network. SCRIPT is a messaging format intended for communication between prescribers and pharmacies as an effort to promote e-prescribing initiatives. I was not able to evaluate the most recent version of SCRIPT, which is available only to NCPDP members or by purchase from ANSI. However, limited documentation regarding *sig* representation in Version 8 is available in the published minutes from a 2005

meeting of the National Committee on Vital and Health Statistics Subcommittee on Standards and Security.[7] The minutes include a description of some of the prescription elements represented in the data standard. These discrete elements include two main fields for quantity and dosage directions. Of note, the dosage directions include a reserved field for an undefined *sig* code and two free-text fields for *sig* instructions, as shown in Table 2.

Table 2: SCRIPT Version 8, documentation of field codes

Field Number	Field Name	Remarks
020-I009	Quantity Composite	This composite is for the count of tablets or number of grams
020-I009-01-6063	Quantity Qualifier	Unit of Measure X-12 DE 355. Values: See External Code List
020-I009-02-6060	Quantity	If Quantity is not submitted, the entire 020-1009 composite is not submitted. Change to an..35 *[sic]* in version 4.0. See sections "Representation" and "Truncation" for syntax and decimal point usage.
020-I009-03-1331	Code List Qualifier	X-12 DE 673. Values: See External Code List
030-I014	Directions	
030-I014-01-7879	Dosage Identification	SIG Code. For future use. Has not been codified yet.
030-I014-02	Dosage	SIG instructions. Dosage free text.
030-I014-03	Dosage	SIG instructions. Dosage free text.

In the absence of complete documentation for Version 10, it is difficult to assess the extent to which the SCRIPT standard formalizes the *sig* in electronic prescriptions. However, the Version 8 documentation suggests that SCRIPT does not attempt to represent *sig* components discretely.

NHS Dose Syntax: Another recent effort is the Dose Syntax project sponsored by the UK National Health Service (NHS) and completed as contracted work by Blue Wave Informatics, LLC. The NHS Dose Syntax project resulted in a final report that was submitted for review in 2005 to the Pharmacy special interest group of Health Level 7 (HL7), an international medical standards organization.[8] The final report is available for download from the Blue Wave Informatics website and is highly recommended reading for those interested in understanding the complexity of representing prescription data in a standard format.[9]

The primary discrete elements of the NHS Dose Model (Appendix A) are:

- Administration Action
- Dose Quantity
    - Quantity Upper Bound
- Timing
    - Frequency
    - Start-Stop
- Route, Site and Method of Administration
- Rate Quantity
- Additional Instructions/Information
- Device Use and Preparation Instructions

The Dose Syntax uses SNOMED-CT as the standard vocabulary for Route and Site of administration. The authors note that while the Method of Administration also requires a standard vocabulary, no corresponding domain exists in SNOMED-CT.

Where applicable the Dose Syntax has explicit mapping of elements to the HL7 Pharmacy Information Model (Appendix B), a subset of the Reference Information Model (RIM). However, the authors explain that the complexity of how timing information is represented in the HL7 v3 RIM impeded their work and, as of the submission of the final report, was therefore incomplete.

In an email correspondence (March 13, 2008), Emma Melhuish, a Pharmaceutical Informatics Specialist with the NHS explains:

> *The Dose Syntax work was carried out by contractors for our organisation and although we received a final report there were still a number of outstanding issues that had not been addressed by the work. Because of the outstanding issues and the complexity of the model there were never any implementations and the work has not been adopted by HL7.*

A web-based testing application developed by Blue Wave Informatics has been taken offline.[10] Nevertheless, the NHS Dose Syntax model represents a superlative attempt to unravel the complexity of prescription *signatura* data and the model served as an excellent reference source for this project.

OpenEHR: The third prescription data model I evaluated is from the OpenEHR project, a not-for-profit initiative jointly sponsored by representatives from the UK and Australia.[5] The foundation of OpenEHR is its library of archetypes. An archetype is a formal expression of a domain content model, based on the

Archetype Definition Language specification. OpenEHR currently maintains hundreds of archetypes for composite **concepts** like "problem list" and "social history", **events** like "procedure" and "transfusion", **observations** like "pulse" and "pupils", and **actions** like "laboratory" and "medication action".[11] The OpenEHR Medication archetype (Appendix C) contains discrete elements for:

- Name of medication
- Administration Instructions
- Strength per dose unit
- Dose unit
- Dose form
- Dose duration
- Route
- Is long term (Boolean value to signify whether the medication is chronic or limited in duration)
- Indication
- Generic Name
- Safety Limits (minimum/maximum weight-based dose, frequency, or interval)
- Administration information
- Dispensing information

Capstone Prescription Data Model: All three models share common elements, specifically a representation of the medication dose. However, all three vary in their modeling of the prescription *signatura*. The NCDCP SCRIPT format contains no discrete *sig* data elements, making it less useful for this project. Both the NHS Dose Syntax and the OpenEHR Medication archetype attempt to define discrete elements of the *sig*, but differ in their choice of modeled attributes. Of

these, the NHS Dose Syntax has broader support for representation of dose ranges, variable timing, and additional instructions.

For this project, I created a hybrid, simplified prescription model that derives many of its elements from the NHS Dose Syntax model, such as the explicit description of dose ranges (Appendix D). The data model for this project included discrete elements for:

- Command (analogous to the NHS Dose Syntax "Action")
- Dose
    - Dose Low
    - Dose High
    - Dose Unit of Measure
- Alternate Dose
    - Alternate Dose Low
    - Alternate Dose Unit of Measure
- Route
- Frequency
    - Frequency Low
    - Frequency High
    - Frequency Unit of Time
- Interval
    - Interval Low
    - Interval High
    - Interval Unit of Time
- Duration
    - Duration Low
    - Duration High

- Duration Unit of Time
- Medication is PRN ("*pro re nata*" or "as needed")

Of note, this model also allows numeric ranges for every numeric value in a script: the dose, frequency, interval, and duration. Consequently, the model could be used to represent a wide variety of *sigs* from extremely simple ("take as needed") to extremely complex ("Take 2 to 4 puffs via nebulizer every 4 to 6 hours for 3-4 days as needed for wheezing"). For development of this prototype parser and for first-round evaluation, I chose not to model some of the NHS Dose Syntax elements, such as compound medication clauses (e.g. "take 2 tabs by mouth every 12 hours for 2 days, then 1 tab by mouth every day"), or preconditions/triggers for medications (e.g. "apply *with every diaper change*" or "take *when BP > 120/90*". Other intentional omissions from the model include numeric ranges on alternate doses, as these were never encountered in the test set and were felt to represent marginal cases. Also, this model does not try to capture the discrete conditions for "PRN" or "as needed" medications. Finally, the model does not try to represent diluents for medications (such as "25mg in 50cc of normal saline").

I.D. Parsing Strategies and Regular Expressions

Methods for automated evaluation of medical free-text data generally fall into two categories 1) keyword or regular expression-based parsers and 2) syntactic/semantic natural language processing strategies. Regular expressions, first developed in the 1950s as a way to classify formal languages, are patterns

that describe a specific set of strings. The patterns can specify optional, alternate, mandatory, or negated content using a straightforward syntax. For example, the Perl regular expression "/H(a|ae|ä)ndel/" would match any of the words "Handel", "Haendel", or "Händel", but not "Haaendel".[12] In this example, the parentheses instruct the Perl regular expression parser to treat the contents as a single group, and the vertical pipe ("|") is an alternation symbol, instructing the parser that any of the three variants are allowed.

Regular expressions and simple pattern matching techniques are ideal for shallow parsing and lexical analysis (for example, generating word frequency counts or creating a lexicon from a corpus of text). Regular expressions are widely implemented in many programming languages such as Perl, Java, Python, and PHP; and generally have exceptionally fast performance.[13]

Regular expressions and pattern matching techniques have been used for a variety of tasks in medical text-mining, such as identification of diabetic, overweight, or hypertensive patients for cohort studies and extraction of blood-pressure information from medical notes.[14,15] In the area of prescription data, previous work at the Brigham and Women's Hospital (Boston, MA) has shown that pattern matching algorithms can be used to help users enter orders in a CPOE system.[16] The Brigham Integrated Computing System (BICS) incorporated a real-time lexical analyzer that attempted to match the words the user was typing against a dictionary of standard terms. Whether or not the match was successful, the user

had the opportunity to correct and augment the results of the computer algorithm, so very high accuracy was not required of the system and therefore its raw performance against a test set of prescription data is unknown. Nevertheless, the system could recognize many common forms and variants of a prescription, such as "Ampicillin 500 mg iv q6h" or "Please give 500 mg of IV ampicillin q 6h".

Kraus *et al* describe a pattern matching approach to finding diabetes-specific drug names and doses in unstructured medical notes.[17] With their algorithm, they were able to achieve a high precision of 97% and an overall recall of 69% to 79%. For individual components of the *sig*, such as the prescribed frequency and route of delivery, the recall was only in the range of 40-60%. The authors propose that further work needs to be done to improve their pattern-based parser. Overall, the performance of regular expression techniques in medical text-parsing is variable and largely influenced by the task complexity.

The more advanced natural language processing (NLP) strategies include syntactic analysis (how words combine to form sentences) and semantic analysis (what words mean). They are computationally more intensive and are more difficult and time-consuming to develop, however they are the better option for analyzing lexically ambiguous text, such as narrative data in medical notes.[13] Meystre and Haug demonstrated the strengths of each strategy in a head-to-head comparison of three NLP parsers, one keyword parser (i.e., regular expression or pattern matching), and a human reviewer.[18] The authors' goal was to extract

medical problems from the narrative text of the Longitudinal Medical Record in use at the University of Utah. In their evaluation, the three NLP parsers achieved a respectable recall of 69 to 89% at the expense of precision, which ranged from 39 to 75%. In contrast, the keyword parser had a precision of 81%, second only to the human reviewers, but it had the lowest recall of 57% (Table 3). Not surprisingly, the regular expression technique was 20 to 40 times faster than the NLP strategies:

Table 3: Comparison of NLP, keyword, and human review strategies for a text extraction task in a corpus of unstructured narrative medical notes.

Measure	NLP 1	NLP 2	NLP 3	Reviewers	Keyword
Recall	0.775	0.892	0.693	0.788	0.575
Precision	0.398	0.753	0.402	0.912	0.807
Time (secs)	72.3	54.7	39	132.2	1.9

Adapted from Meystre and Haug, 2005. 95% confidence intervals are not shown.

Finally, it is worth noting that commercial NLP prescription parsers do exist, such as FreePharma (Language & Computing, Inc. Bethesda, MD). FreePharma has been successfully used at the Marshfield Clinic to identify research subjects with diabetes mellitus by identifying mentions of any of three different classes of glucose-lowering medications.[19] This system was excluded from consideration for this project because of the cost of the commercial parser.

Prescription data is unique in that it represents a densely abbreviated, semi-structured form of medical text. This is similar to the "notational text" described by Barrows and Friedman in short-form medical notes, such as surgical progress notes.[20] The authors give an example of notational text that requires extensive

lexical knowledge to decipher: "S/B Cx (+) GPC c/w PC, no GNR", which translates to "sputum and blood cultures were positive for gram-positive cocci consistent with pneumococcus. No gram-negative rods". In comparison, one could easily encounter an ambulatory pediatric *sig* with just as many abbreviations: "2ml via GJ QAC & QHS PRN agitation", which translates to "2 milliliters given via the gastro-jejunal tube before each meal and at bedtime as needed for agitation". While Barrows & Friedman demonstrated that the MedLEE NLP parser was comparable to pattern matching techniques for notational text, they do not comment on the relative speed of the two techniques and do not suggest that NLP is the preferred strategy for notational text. Instead, they express surprise that MedLEE performed as well as it did.

The existing literature suggests that regular expression-based algorithms can be used to parse semi-structured data with accuracy sufficient for use by clinical researchers. Because prescription data are more like "notational text" than not, I felt that regular expressions were a justifiable approach to *sig* parsing. However, efforts to date have not specifically been tested against pediatric prescription data, and most studies have focused on identifying a handful of key tokens (such as medication names or numeric doses in narrative text). Therefore, current strategies are not generalizable for large sets of pediatric prescription data or complex *sig* models with more than a few tokens. Based on my review of the literature, I chose a pattern matching / regular expression technique over semantic/syntactic NLP techniques for the following reasons:

1) In theory, pattern matching should perform better-than or as-well-as NLP for notational text.
2) Regular expressions are easier to program than NLP algorithms.
3) I was familiar with regular expressions from prior programming work, whereas learning NLP techniques specifically for this project would have been prohibitively time consuming.
4) Regular expression algorithms are significantly faster than NLP, which is a relevant consideration for researchers who want to use the tool to mine massive datasets, or if the parser were to be implemented in a real-time system.

II. Materials and Methods

II.A. Data Retrieval

The data retrieval and analysis for this study was conducted with a "Not Human Subjects" exemption from the CHOP Institutional Review Board (Application #5828), granted on February 15, 2008. The CHOP pediatric network includes nearly 40 ambulatory clinics across the Delaware Valley; a number of specialty care centers and surgical centers; and a tertiary care, 380-bed urban teaching hospital located in Philadelphia.

All of the ambulatory primary care sites use EpicCare as their CPOE system and health record. The EpicCare application conducts all transactions via a Caché database instance known as "Chronicles". Each night, there is a data extraction process that pushes new data from Chronicles into a relational database instance named "Clarity", implemented as an Oracle database. Using Oracle SQL Developer version 1.2.1 (Oracle Corporation, Redwood Shores, CA), I obtained roughly 11,500 free-text prescriptions with associated data fields to indicate the therapeutic class of the medications, the name and formulation of the medication, and the start/stop date of the medication, where available. The SQL query was written to ensure that I retrieved a random sample of the over 2-million prescriptions in our system (Figure 1).

Figure 1: Orace SQL statement used to extract prescriptions from Clarity schema

```
select c.title, a.order_med_id, a.medication_id, a.description,
   a.sig, a.dosage, a.quantity, a.refills, a.start_date,
   a.end_date, b.GPI
from (select * from order_med sample(0.5)) a,
   clarity_medication b, zc_thera_class c
where a.MEDICATION_ID  = b.MEDICATION_ID
   and b.thera_class_c = c.thera_class_c
   and b.gpi is not null
   and a.sig is not null
order by order_med_id;
```

The extracted data set was saved as a comma-separated text file in Microsoft Excel 2004 for Mac, version 11.3.5 (Microsoft Corporation, Redmond, WA). At a later date, I exported increasingly larger sets of data (from 20,000 to 360,000 scripts) for throughput testing of the RxParse module. Ten thousand *sigs* from the original set were used to create a word frequency map. A subset of 1,000 was used for the final parser evaluation.

II.B. Algorithm Development

The RxParse prescription parsing module was developed on a MacBook Pro laptop computer with a 2.4 GHz Intel Core 2 Duo processor and 4 GB of RAM, running Mac OS X, version 10.4.11 (Apple Inc., Cupertino, CA). RxParse is written in Perl version 5.10.0, using the ActivePerl distribution (ActiveState Software, Inc., Vancouver, BC). This version was chosen over earlier releases of Perl because of enhanced support for specific regular expression functions such as named capture buffers. Named capture buffers allow the parser to assign an explicit name to a matched pattern. In this way, the parser can identify a complex

pattern, such as a numeral followed by a unit of measure, and assign the numeral to one variable while assigning the unit of measure to another. This feature is used extensively in the parsing algorithm. As a result of this design choice, RxParse will not work under earlier versions of Perl and will elicit a syntax error at runtime. I used the Eclipse integrated development environment version 3.3.0 (Eclipse Foundation Inc., Ottowa, Ontario, Canada) and the EPIC Perl IDE plugin version 0.6.2.2 (available at http://e-p-i-c.sourceforge.net/) for syntax checking and debugging.

The first phase of design consisted of describing the anticipated behavior of RxParse using a set of use cases. One particular use case (Table 4) highlights a number of key specifications for the parser. **First,** the parser should be able to recognize numbers described as text as well as numerals (both "one" and "1"). **Second,** the parser should recognize text fractions ("one-half"). **Third,** RxParse should automatically recognize and expand Latin abbreviations, regardless of case or punctuation ("P.O." = "po" = "by mouth"). **Fourth,** any unrecognized fields should be reported as "null". **Fifth**, all commands, units of measure, routes, and units of time, should be mapped to their canonical forms ("tabs" = "tablet", "days" = "day"). These canonical forms should, in turn, correspond to a standard lexicon derived either locally at CHOP or, where available, from existing standard terminologies.

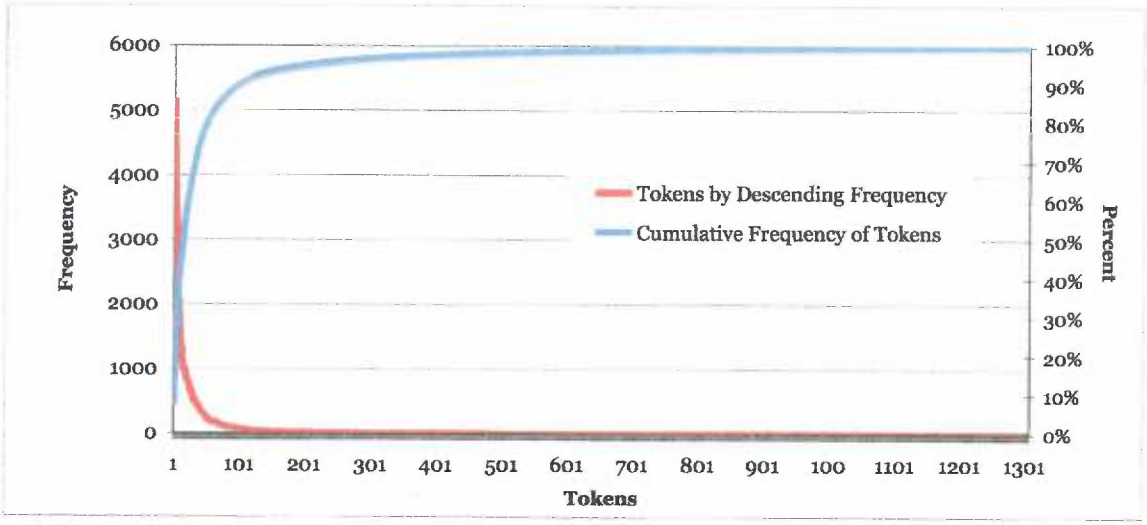
Table 4: Use case showing the expected output of the parser given a free-text *sig*

INPUT	Take one and one-half tab (250mg) P.O. every 4-6 hours for 2 days	
OUTPUT	**CMD**	take
	DoseLow	1.5
	DoseHigh	null
	DoseUOM	TABLET
	AltDoseLow	250
	AltDoseUOM	mg
	Route	oral
	FreqLow	null
	FreqHigh	null
	FreqUOT	null
	IntLow	4
	IntHigh	6
	IntUOT	hour
	DurLow	2
	DurHigh	null
	DurUOT	day
	IsPRN	FALSE

Word Frequency Analysis: To derive a local vocabulary for the canonical forms of various *sig* components, I first completed a word-frequency analysis on a subset of 10,000 prescriptions using a small Perl script. The full set of 10,000 scripts contained nearly 69,791 individual tokens, or an average of about 7 tokens per script. Within this set, there were 1,308 distinct tokens with an average frequency of 53 occurrences in the dataset. The top 15 tokens represent 50% of all tokens, and the top 50 tokens represent 80% of all tokens. The 150 most-frequently occurring tokens in the dataset are shown in Appendix E. Figure 2 shows a plot of the cumulative and total frequencies of all 1,308 tokens. While the vast majority of words in the dataset are represented by just the first 150 tokens, the plot shows a very long tail, implying that many hundreds of words,

spelling variations, and abbreviations occur rarely, even in this large dataset of 10,000 scripts.

Figure 2: Word Frequency Analysis



Command: Using the word frequency list and manually refining it with subsequent test sets, I constructed a local vocabulary to describe the "Command" component of the *sig* model (Table 5). The list includes common words as well as variants (dispense, dispence) and misspellings (aply, aplly, apply, appply). By combining these terms with the regular expression "vertical pipe" alternation operator, the parser looks for any of the terms in the *sig*. Each term in the vocabulary, including the misspelled terms, are mapped to a canonical form by the parser.

Table 5: Standard Vocabulary for "Command"

Standard Vocabulary describing "Command" Element			
add	dispence	phone	wash
administer	dispense	place	wipe
aply	disolve	pull	
aplly	dissolve	put	
applu	dress	repeat	
apply	drink	replace	
appply	fill	resume	
avoid	finish	rinse	
begin	flush	rub	
brush	give	shampoo	
change	include	spray	
check	increase	sprinkle	
chew	inject	swallow	
coat	insert	swish	
contact	instill	take	
continue	keep	tape	
cover	leave	taper	
crush	massage	treat	
dilute	mix	use	
discard	pack	wait	

Dose: Following the NHS model, a dose was described as a number or range followed by a unit of measure. The parser was designed to accommodate many types of numbers, including text numbers, numerals, fractions, and even punctuated numbers such as "1,000,000" for 1 million. The standard vocabulary for units of measure and their canonical forms were derived from the "ZC_MED_UNIT" table in the EpicCare Clarity schema. Some local additions were added to accommodate common units of measure in use at CHOP, such as a "gummy" (a type of chewable candy multivitamin, as in "take 1 gummy by mouth daily") and "neb" (short for "nebulized respiratory treatment"). Also, the parser was designed to accept common misspellings seen in the word frequency analysis, such as "teapoon" for "teaspoon". As in the "Command" component, misspellings and alternate forms are later mapped to their canonical forms (Table

6). Also as with the "Command" component, the dose units of measure (DoseUOM) were combined with the regular expression alternation operator such that the parser would match a number or range followed by any of the units of measure in the list. The regular expression for alternate or parenthetical dosage forms was implemented in the same way.

Table 6: Dose Unit of Measure Examples

Sample of DoseUOM Vocabulary and Canonical Form	
Term	**Canonical Form**
'act'	ACTUATION
'amp'	AMPULE
'ampule'	AMPULE
'applicator'	APPLICATOR
'bar'	BAR
'bottle'	BOTTLE
'c'	CAPSULE
'can'	CAN
'cap'	CAPFUL or CAPSULE
'capsule'	CAPSULE
'cc'	CUBIC CENTIMETER
'chew'	CHEWABLE TAB
'chew tab'	CHEWABLE TAB
'chew tablet'	CHEWABLE TAB
'chewable'	CHEWABLE TAB
'chewable tab'	CHEWABLE TAB
'chewable tablet'	CHEWABLE TAB

Route: The U.S. Federal Drug Administration Center for Drug Evaluation and Research maintains a data standard manual with a list of accepted medicinal routes, their definitions, canonical short forms, FDA code, and NCI Concept IDs. Because the list is comprehensive (there are 111 standard routes ranging from AURICULAR (OTIC) to VAGINAL), this list and its mappings were adapted for use by RxParse. The list had to be heavily modified to accommodate the abundant abbreviated forms and variations of common routes. An example of

these modifications for the route "NASAL" is shown in Figure 3. The code sample also demonstrates how mappings of variants to canonical forms and standard terminologies were implemented in Perl using the "hash of hashes" construct (a.k.a. multidimensional array).

Figure 3: Perl code showing the mapping of terms to FDA route "NASAL"

```
foreach my $element ('nasal' , 'intranasal' , 'nostril' , 'nare' ,
'naris')
{
        $Route{$element}{'name'} = 'NASAL';
        $Route{$element}{'def'} =
          'Administration to the nose; administered by way of the nose.';
        $Route{$element}{'fda'}  = 'NASAL';
        $Route{$element}{'code'} = '14';
        $Route{$element}{'nci'}  = 'C38284';
}
```

Frequency, Interval, and Duration: All three of the time-based components of the data model required a standard terminology for units of time. However, because I could find no such list, I constructed one from the word frequency analysis and augmented it through serial testing. Similar to the route elements, units of time and their common misspellings and variants were mapped to a canonical form using multidimensional arrays. In addition, because one of the objectives was to give researchers the ability to perform subsequent calculations on the parsed values, each text unit of time was mapped to a "daily equivalent" and an "hourly equivalent" where possible. For example, the term "week" can be used in a frequency ("3 times per week"), in an interval ("every 2 weeks"), or in a duration ("for 1 week"). Mapping the term to numeric equivalents allows users to use the parsed values in calculations, as shown in Table 7.

Table 7: Examples of derived calculations from unit of time mappings

Term	Daily Equivalent (DE)	Hourly Equivalent (HE)	Usage	Example Calculation
week	7 days	168 hours	Duration = "3 weeks"	Total duration = (3 x DE) = 21 days
hour	1/24 days	1 hour	Interval = "once every 4 hours"	Number of daily doses =1 / (4*DE) =6 doses

Figure 4 shows how the common variants, abbreviations, and misspellings for the unit of time "WEEK" were implemented in Perl and mapped to the numeric daily equivalent, hourly equivalent, and canonical form.

Figure 4: Implementation of Unit of Time mappings in Perl

```
foreach my $unit ('week' , 'weekly' , 'wk' , 'weks' , 'wkly' , 'aweek')
{
        $UOT{$unit}{'de'} = 7;
        $UOT{$unit}{'he'} = 168;
        $UOT{$unit}{'cf'} = 'WEEK';
}
```

The three time-based components had slightly different regular expressions to describe them. For example, a frequency was any string that matched a fairly complex pattern including:

1) **optional** preceding phrase "up to"
2) **mandatory** number or range of numbers
3) **mandatory** word ("x" OR "time" OR "times")
4) **optional** word ("a" OR "per" OR "each")
5) **mandatory** unit of time
6) **optional** suffix "ly"

As a result, the parser can recognize a large variety of phrases as valid frequencies. Examples include "up to 3 x per day", "1-2 times each morning", "1 time a month", or simply "twice weekly".

The implementation of this regular expression in Perl is shown in Figure 5. Because they are difficult to read without an understanding of Perl regular expression syntax, the mandatory and optional phrases are highlighted for clarity.

Figure 5: Perl regular expression for prescription frequency

```
$Sig{frequency} = qr/
      (?:
            (?:up\sto)?\W*
            (?<F1>$Sig{_number})\W*
            (?:
                  (?:
                        (?:to|\-)\W*
                        (?<F2>$Sig{_number}\W*)
                  )
            )?
            (?:x|times?)\W*(?:a|per|each)?\W*
            (?<FU> $Sig{_uot} )(?:ly)?\b
            (?{ ( $_{FREQ1} , $_{FREQ2} , $_{FREQuot} ) =
      ( $+{F2} ) ? ( $+{F1} , $+{F2} , $+{FU} ) : ( $+{F1} , "" ,
$+{FU}); })
      )
      |
      (?:
            (?:a|per)?\W*
            (?<F>$Sig{_uot})(?:ly)\b
            (?{ ( $_{FREQ1} , $_{FREQ2} , $_{FREQuot} ) =
                  ( "1", "" ,$+{F} ); })
      )
/ix;
```

The code in Figure 5 also shows how regular expressions can be combined into increasingly complex expressions. The regular expression for "frequency" includes a reference to a previously-defined regular expression for a generic number ("$Sig{_number}" in line 4 of the code sample). The regular expressions for "interval" and "duration" are constructed similarly and vary only in their mandatory and optional text patterns.

IsPRN: To determine if a prescription is to be given on an "as needed" or "prn" basis, I used a simple regular expression pattern that looked for:

1) **mandatory** phrase ("only" OR "if needed" OR "as needed" OR "prn")
2) **optional** word "for"

Complete *Sig*: By combining the regular expressions for each of the above components, the parser defines a *sig* as one of two standard sequences that vary only in the position of the route.

- Sig Form One:
  1) optional action
  2) optional dose
  3) optional route *(before interval/frequency)*
  4) optional (interval OR frequency)
  5) optional duration
  6) optional isPRN

- Sig Form Two:
  1) optional action
  2) optional dose
  3) optional (interval OR frequency)
  4) optional route *(after interval/frequency)*
  5) optional duration
  6) optional isPRN

In other words, the parser would recognize either "give 1 tab by mouth every 4 hours" or "give 1 tab every 4 hours by mouth" as valid, complete sigs. By

making each element optional, the parser specifies the order of the elements, but not their presence. Therefore, the parser would also successfully parse the sig "give every 4 hours as needed" despite the fact that it is missing a dose, route, and duration.

II.C. Validation Plan

For this project, two human judges (the author and his wife) validated the performance of the RxParse algorithm. Validation took place in two phases. First, the judges independently parsed a set of 200 prescriptions from which inter-rater reliability could be calculated. Inter-rater reliability was calculated using Cohen's kappa coefficient, which corrects for the fact that judges may agree or disagree by random chance.[21] The kappa statistic thresholds proposed by Landis and Koch were used to set an acceptable level of inter-rater reliability as any kappa value greater than 0.60.[22] Landis and Koch's threshold values are shown in Table 8.

Table 8: Kappa threshold values adapted from Landis & Koch

Kappa	Agreement
<0	Less than chance agreement
0.01-0.20	Slight agreement
0.21-0.40	Fair agreement
0.41-0.60	Moderate agreement
0.61-0.80	Substantial agreement
0.81-0.99	Almost perfect agreement

Second, the judges scored the performance of the parser's output for 1,009 randomly selected scripts in the CHOP dataset. Each judge validated roughly 500 scripts in a variety of medication classes. The hope was that by scoring such a large number of scripts, we could assess not only the parser's overall

performance, but also its performance within specific classes of medications. The parser's performance was evaluated for sensitivity, specificity, positive predictive value, and negative predictive value with 95% confidence intervals.

If the human judge is the "gold standard", the parser is akin to a diagnostic test. Each component of the *sig* represents a state whose presence or absence one is trying to predict using the test. The goal is to evaluate the test's ability to discern the state (presence or absence of a *sig* component).

The agreement or disagreement between algorithm and human can be described using a standard 2x2 table (Table 9):

Table 9: Standard 2x2 contingency table comparing a test and gold standard

	Human Positive	**Human Negative**	**Total**
Algorithm Positive	TP	FP	TP+FP
Algorithm Negative	FN	TN	FN+TN
Total	TP+FN	FP+TN	TP+FN+FP+TN

TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

Imagine the scenario in which 10 prescriptions are being evaluated for the presence of a valid *sig* route. The human reviewer finds that 8 of the 10 have a valid route, of which the algorithm detects 7 and misses 1. The human finds that 2 of the 10 do not have a valid route, and the algorithm determines that the same 2 have a missing route. We can fill in the 2x2 table with these data (Table 10):

Table 10: Example contingency table

Component: ROUTE	Human Positive	Human Negative	Total
Algorithm Positive	7	0	7
Algorithm Negative	1	2	3
Total	8	2	10

From these values, we can calculate that the sensitivity of the algorithm is 7/8, the specificity is 2/2 and the overall accuracy is 9/10. Furthermore, the positive predictive value is 7/7, and the negative predictive value is 2/3.

This set of calculations was repeated for each of the 17 *sig* components. Of note, however, simply because a component parses correctly does not mean the parent concept was successfully parsed. For example, a complete "dose" could include up to four distinct sub-components (DoseLow, DoseUOM, AltDose, and AltDoseUOM). In the context of real-world use, a parser that – for example – only identifies the unit of measure but fails to identify a valid dose is not ideal. Nevertheless, for the purposes of a first-round validation, it was necessary to validate each component separately, as the process would direct future enhancements to the parser. All calculations were performed using an online clinical statistics application available through Vassar College.[23]

A third level of validation was performed out of curiosity. Early testing showed that the parser was remarkably fast, handling 10,000 prescriptions in a matter of seconds. To test the throughput of the parser and to quantify the effect of increasingly larger datasets on the parser performance, I used a high-resolution timer module written in Perl by Jarkko Hietaniemi named Time::HiRes, which is

available via the Comprehensive Perl Archive Network.[24] The parser's throughput was tested across a range of datasets from 10,000 to over 350,000 prescriptions.

III. Results

III.A. Measurement of Inter-Rater Reliability

Two human judges (the author, a general pediatrician; and his wife, a pediatric oncologist) were assigned the task of validating the performance of the RxParse Perl module. Both have practiced ambulatory pediatrics in a primary care setting during their residency training and interact with *sig* data on a near-daily basis. To ensure that the judges were scoring the performance of the parser consistently, a study of inter-rater reliability was conducted using a sample set of 200 raw scripts from the CHOP EpicCare dataset.

Each judge was given an Excel spreadsheet that contained the 200 scripts along with definitions of each of the 17 components of the prescription data model. For each of the 200 scripts, the judges were asked to identify any occurrences of the 17 elements in the script. Judges were asked to leave a field blank if the element was missing from the script. Once complete, the two sets of human-parsed *sigs* were compared and scored for concordance. Cohen's kappa coefficient was calculated for each of the 17 components, with an acceptable threshold set at a kappa value of 0.6 or greater.

Despite the densely abbreviated and notational style of prescription *sigs*, the ability of two independent judges to parse them was remarkably consistent (Table 11). Kappa coefficients for all 17 data elements were exceptional, ranging from

0.71 ("substantial agreement") to 1.00 ("perfect agreement"). Based on the high correlation between the two judges, we felt comfortable proceeding with the parser validation, allowing each judge to score approximately 500 *sigs* without cross-validation by the other judge.

Table 11: Inter-rater reliability scores as measured by Cohen's kappa

Element	Both +	Both -	Judge 1 + Judge 2 -	Judge 2+ Judge 1 -	Observed agreement	Chance agreement	Kappa
Command	44	143	5	8	0.94	0.62	**0.83**
DoseLow	152	46	0	2	0.99	0.64	**0.97**
DoseHigh	8	192	0	0	1.00	0.92	**1.00**
DoseUOM	147	47	0	6	0.97	0.63	**0.92**
AltDose	14	188	0	0	1.00	0.87	**1.00**
AltDoseUOM	14	188	0	0	1.00	0.87	**1.00**
Route	179	12	9	0	0.96	0.85	**0.71**
FreqLow	104	89	2	0	0.99	0.50	**0.98**
Freqhigh	3	197	0	0	1.00	0.97	**1.00**
FreqUOT	105	90	2	0	0.99	0.50	**0.98**
IntLow	60	137	1	2	0.99	0.57	**0.97**
IntHigh	10	188	2	0	0.99	0.90	**0.90**
IntUOT	60	136	2	2	0.98	0.57	**0.95**
DurLow	55	141	4	0	0.98	0.59	**0.95**
DurHigh	70	127	2	1	0.99	0.54	**0.97**
DurUOT	4	195	0	1	1.00	0.96	**0.89**
IsPRN	71	127	1	1	0.99	0.54	**0.98**

III.B. Parser Evaluation

Accuracy: Table 12 shows the summarized performance of the parsing algorithm as judged by the two human judges. The prevalence of individual components ranged from about 1% (Frequency High) to over 80% (Route) within the 1009 scripts that were reviewed. The sensitivity of each component ranged from 0.33 (Frequency High) to 1 (Dose High). Perhaps because of the explicit pattern-

Table 12: Parser performance for each *sig* component

Component	True +	True -	False +	False -	Prevalence (95% CI)	Sensitivity (95% CI)	Specificity (95% CI)	PPV (95% CI)	NPV (95% CI)
CMD	243	758	0	8	0.25 (0.22-0.28)	0.97 (0.94-0.99)	1 (0.99-1)	1 (0.98-1)	0.99 (0.98-1)
DoseLow	777	215	2	15	0.78 (0.76-0.81)	0.98 (0.97-0.99)	0.99 (0.96-1)	0.99 (0.98-1)	0.93 (0.89-0.96)
DoseHigh	29	980	0	0	0.03 (0.02-0.04)	1 (0.85-1)	1 (0.99-1)	1 (0.85-1)	1 (0.99-1)
DoseUOM	779	217	2	11	0.78 (0.76-0.81)	0.99 (0.97-0.99)	0.99 (0.96-1)	0.99 (0.99-1)	0.95 (0.91-0.97)
AltDoseLow	109	897	0	3	0.11 (0.09-0.13)	0.97 (0.92-0.99)	1 (0.99-1)	1 (0.96-1)	0.99 (0.99-1)
AltDoseUOM	109	898	0	2	0.11 (0.09-0.13)	0.98 (0.93-0.99)	1 (0.99-1)	1 (0.96-1)	0.99 (0.99-1)
Route	641	194	1	173	0.81 (0.78-0.83)	**0.79 (0.76-0.81)**	0.99 (0.97-1)	0.99 (0.98-1)	**0.53 (0.48-0.58)**
FreqLow	437	472	0	100	0.53 (0.50-0.56)	**0.81 (0.78-0.84)**	1 (0.99-1)	1 (0.99-1)	**0.82 (0.79-0.85)**
FreqHigh	3	1000	0	6	0.01 (0.00-0.02)	**0.33 (0.09-0.69)**	1 (0.99-1)	1 (0.31-1)	0.99 (0.99-1)
FreqUOT	440	472	0	97	0.53 (0.50-0.56)	**0.82 (0.78-0.85)**	1 (0.99-1)	1 (0.99-1)	**0.83 (0.80-0.86)**
IntLow	263	717	1	28	0.29 (0.26-0.32)	0.90 (0.86-0.93)	0.99 (0.99-1)	0.99 (0.98-1)	0.96 (0.95-0.97)
IntHigh	67	936	0	6	0.07 (0.06-0.09)	0.92 (0.82-0.97)	1 (0.99-1)	1 (0.93-1)	0.99 (0.98-1)
IntUOT	263	720	1	25	0.29 (0.26-0.31)	0.91 (0.87-0.94)	1 (0.99-1)	1 (0.98-1)	0.97 (0.95-0.98)
DurLow	220	702	0	87	0.30 (0.28-0.33)	**0.72 (0.66-0.77)**	1 (0.99-1)	1 (0.98-1)	**0.89 (0.87-0.91)**
DurHigh	8	992	0	9	0.02 (0.01-0.03)	**0.47 (0.23-0.71)**	1 (0.995-1)	1 (0.60-1)	0.99 (0.98-1)
DurUOT	220	702	0	87	0.30 (0.28-0.33)	**0.72 (0.66-0.77)**	1 (0.99-1)	1 (0.98-1)	0.89 (0.87-0.91)
IsPRN	201	757	0	51	0.25 (0.22-0.28)	**0.80 (0.74-0.84)**	1 (0.99-1)	1 (0.98-1)	0.94 (0.92-0.95)
TOTAL	**4809**	**11629**	**7**	**708**	**0.32 (0.31-0.33)**	**0.87 (0.86 – 0.88)**	**0.999 (0.999-1)**	**0.998 (0.997-0.999)**	**0.943 (0.938-0.947)**

*Values less than 0.85 highlighted in gold

matching inherent to regular expressions, specificity was universally high, consistently in the range of 0.99 to 1. Like specificity, the positive predictive value (PPV) is related to the number of false positives. Because false positives were so rare in the validation set, the PPV for each component was also very high, in the range of 0.99 to 1. In other words, given that the parser found an element, the user could be assured that the parsed component was accurate. In contrast, the negative predictive value (NPV) is linked to the overall number of false negatives. Not surprisingly, parsed components with high rates of false negatives (Route, Frequency Low, Frequency Unit of Time, and Duration Low had somewhat lower negative predictive values. In other words, for those components, the absence of a parsed value in the result set could not rule out the presence of the component in the original *sig*. When the performance statistics across all 17 components were aggregated, RxParse had a sensitivity (recall) of 0.87 and a positive predictive value (precision) of over 0.99.

Throughput: The completed RxParse algorithm contains over 1800 lines of code. The compiled master regular expression for a valid prescription, with every possible alternation, substitution, and variation, is over 400 lines long. Because of the length and complexity of the regular expression, I was interested in testing the speed and throughput of the parser. I obtained data extracts from EpicCare ranging in size from 11,075 scripts to 357,097 scripts. For each dataset, I used the Time::HiRes Perl module to obtain precise measurements of the total runtime of

the algorithm, excluding the time to load the dataset into memory since this would be highly variable in a production implementation.

Each dataset was timed for 10 complete runs. Both the time to completion in seconds as well as the scripts parsed per second were recorded for each run. For each dataset, I obtained the mean, standard deviation, and 95% confidence intervals. The results of the timed trials for each dataset are shown in Table 13 and Table 14. Of note, the increase in elapsed time is very nearly linear, implying no decrement in performance with progressively larger datasets within the ranges tested. On closer examination of the scripts parsed per second, there is a slight but noticeable decrease in the number of scripts parsed per second with larger datasets.

Table 13: Average elapsed time for parsing 6 datasets, 10 trials each

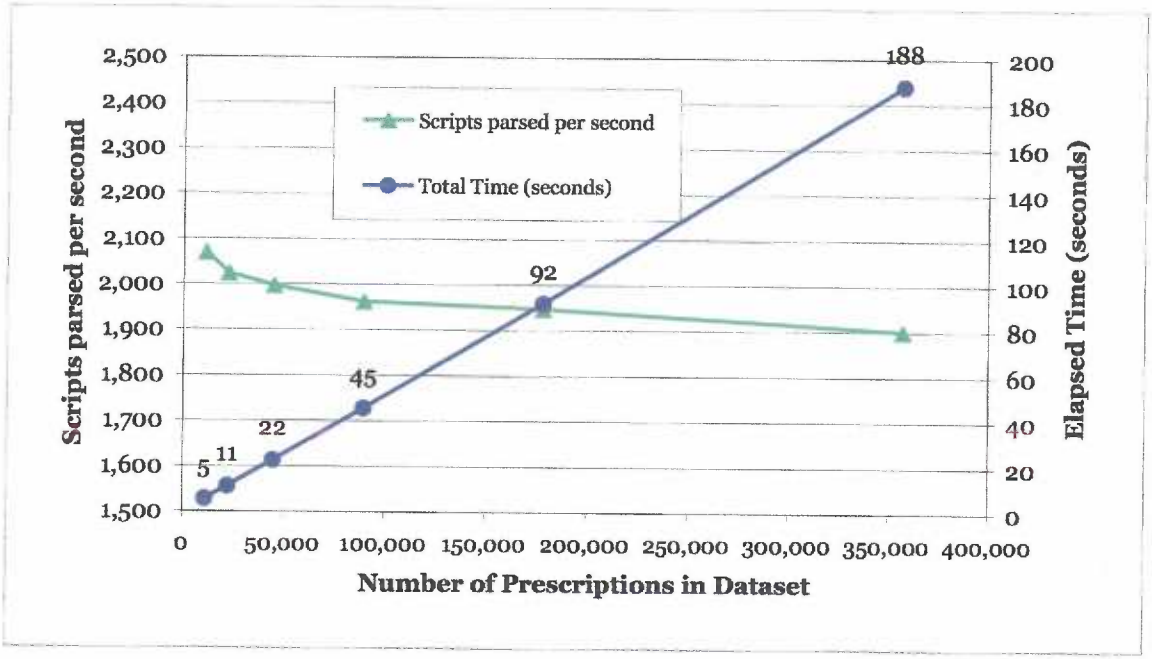
Total Scripts Parsed	Elapsed Time (secs)	Standard Deviation	95% CI (secs)
11075	5.36	0.27	5.19 - 5.53
22444	11.10	0.33	10.89 - 11.3
44820	22.45	0.35	22.23 - 22.67
89275	45.49	0.44	45.22 - 45.76
178549	91.69	0.62	91.31 - 92.07
357097	187.86	0.47	187.57 - 188.15

Table 14: Average scripts parsed per second for 6 datasets, 10 trials each

Total Scripts Parsed	Scripts parsed per second	Standard Deviation	95% CI (scripts/sec)
11075	2069.75	94.36	2011.26 - 2128.23
22444	2023.81	61.02	1985.99 - 2061.63
44820	1996.88	31.55	1977.33 - 2016.44
89275	1962.64	18.71	1951.04 - 1974.23
178549	1947.39	12.93	1939.37 - 1955.41
357097	1900.87	4.78	1897.90 - 1903.83

For the largest dataset (over 350,000 prescriptions), the number of scripts parsed per second was approximately 1,900 per second, about 100 scripts per second less than for the smallest dataset of only 11,000 scripts. These trends are shown graphically in Figure 6. Despite the slight decrement in parsed scripts per second with large datasets, it is still feasible that RxParse could parse every electronic prescription in the CHOP Ambulatory network – over 2 million *sigs* – in a mere 20 minutes.

Figure 6: Parser performance as scripts per second and elapsed time



IV. Discussion

Overall, the parser performed significantly better than expected. One measure of the parser's performance is its overall accuracy defined as the percent of scripts for which there were no parsing errors, either false positives or false negatives, in any of the 17 components. By this definition, the parser had an overall accuracy of 69.2% (698 of 1009 scripts). On average, the parser returned the correct answer for 16.3 of the 17 components (96%).

Based on tallies of all true and false positive and negative results for all 17 components, the parser had an overall sensitivity of 0.87 (95% CI: 0.86-0.88), specificity of 0.999 (95% CI: 0.999-1.000), positive predictive value of 0.998 (95% CI: 0.997-0.999), and negative predictive value of 0.943 (95% CI: 0.938-0.947). These results as well as the performance for each individual component are summarized in Table 12.

For this first version of a general-purpose prescription parser based on regular expressions, I hoped to attain sensitivity (recall) and positive predictive values (precision) that were similar to the results of Kraus *et al*,[17] who were able to achieve an overall sensitivity of 69-79% with a parser designed to identify a subset of medications for diabetes mellitus. However, there are significant differences between RxParse and the Kraus parser that make a direct comparison difficult. First, the Kraus parser was designed to identify medication data in

narrative notes, while RxParse is designed for individual free-text *sigs*. Second, the Kraus parser identifies four elements (drug name, dose, route, and frequency) while RxParse uses a more complex model that includes 17 elements. The Command, Dose, Alternate Dose, and Interval elements all had very high recall and precision, all greater than 97%. The Route, Frequency, and Duration components tended to have lower sensitivity, higher false negative rates, and lower negative predictive values. Evaluation of the false negatives for these three components was revealing:

Route: For many medication scripts, the route of administration is explicitly stated ("take 2 tabs by mouth every 4 hours"). However, it is not uncommon to imply the route for common medications (e.g.: "take 2 tabs every 4 hours"). Because the parser does not attempt to guess the route, all implied routes are missed. Similarly, many prescriptions for respiratory medications have an implied route, usually discernable from the unit of measure (e.g.: "take 2 puffs every 4 hours" or "1 neb every 4 hours"). Another major class of missed routes was for topical medications, where dosing instructions can take myriad forms. One script with a false-negative route read "apply to clean foot every other day". While a human reviewer can discern that applying something to the foot is consistent with a topical medication, the regular expression cannot.

A better strategy for implied routes is perhaps to obtain clues from the medication name or therapeutic class itself. Medications like "triamcinolone ointment" (a

topical steroid) are by definition topical medications. Given a large enough drug-name lexicon, future versions of RxParse could look for an implied route based on a lookup table of medication names or classes. Another strategy would be to pair certain dose units of measure with certain routes. For example, "nebs" and "puffs" could be linked to a respiratory route, or "squirts" could be limited to the nasal route. Of the 173 scripts with false negative routes (17% of all prescriptions), the vast majority were due to topical (88 of 173 = 51%) and inhaled medications (56 of 173 = 32%), indicating this is an area of future work.

Frequency: The linear nature of the parser and, specifically, the large regular expression used to parse valid *sigs* has the drawback of inflexibility. The parser expects sets of words in specific orders, and does not allow for "extra" words. Unfortunately, topical medication *sigs* are highly variable. An example would be "apply a thin coat to affected area four times daily". Though this script has a valid command ("apply) and frequency ("four times per day"), the parser does not recognize the 6 words between the command and the frequency, and therefore cannot identify the frequency. Of the approximately 100 false-negative frequencies and 97 false-negative frequency units of time, 62-63% were again due to the less structured nature of topical medication prescriptions (63 of 100 frequencies and 60 of 97 units of time, respectively).

A straightforward modification to the parser might help overcome this: rather than mandating the sequence of dose components (and risking that the parser fails

early in the string due to unrecognized words), the parser could be redesigned to allow a component pattern to be matched anywhere in a string. This might also enable the algorithm to parse compound prescriptions, like "take 2 tabs every day for 2 days, then 1 tab every other day for 1 week, then stop". By allowing multiple matches of a component anywhere in the string, the parser could reconstruct each distinct sig within a compound sig.

Duration: Two classes of medications contributed to the poorer performance of the duration parsing algorithm. First, many nebulized respiratory treatments ordered in the ambulatory setting are intended to be given for acute wheezing in the office. These medications, unlike respiratory medications commonly given via metered dose inhaler at home, are usually in the form of albuterol mixed with a normal saline diluent. For example, the script may read "0.5ml neb in 3ml saline x 1 dose, now". Because the parser's dose algorithm does not recognize diluents, all subsequent components after the dose will fail. The parser recognizes "0.5" as the dose, "ml" as the unit of measure, and "neb" as the route, but then stops when it reaches the saline diluent. This limitation could similarly be overcome by modifying the algorithm such that it matches any component anywhere in the string, regardless of order or position. The second class of medication that contributed to the high false negative rate for the duration component was topical preparations for reasons again related to the parsers inflexibility in the face of less-structured prescriptions.

Effect of Therapeutic Class: RxParse performed well on anti-infective and analgesic medications. Although a sub-analysis of the parser's performance by therapeutic class was not conducted, the task would prove useful and is planned for future versions of the parser. Antibiotic scripts, though sometimes complex, tend to be highly structured in their syntax: "take 250mg (5ml) by mouth every 4 hours for 7 to 10 days". As a result, it is not unreasonable to assume that the parser would excel at the task of parsing this class of medications.

Study Limitations and Next Steps: Because testing and validation was performed on a pediatric dataset, it is likely that RxParse will perform differently on an adult prescription dataset. Also, because the word frequency analysis helped to define the local lexica for some of the components, the parser may be over-fit to CHOP data. Evaluating these limitations would require cross-validation against non-CHOP ambulatory prescription data. In a subsequent project, we hope to expand on the work begun in this Capstone by validating RxParse using a set of prescriptions from the Kaiser Permanente Northwest ambulatory network. This would allow us to test the parser on a set of adult prescription data. Like CHOP, Kaiser also uses the EpicCare ambulatory CPOE system.

Finally, this early work demonstrates that RxParse performs better at parsing certain medication classes (e.g. antibiotics) than others (e.g. topical medications).

A formal sub-analysis would allow clinical researchers to decide whether RxParse is the correct tool to use based on their research question.

Overall, I was very pleased with the development and validation of RxParse. The tool, even in these early stages, shows promise in terms of its ability to rapidly extract meaningful data from a corpus of ambulatory prescriptions. With additional development, the tool can be improved to overcome some of the existing problems in the algorithm, specifically with respiratory and topical medications.

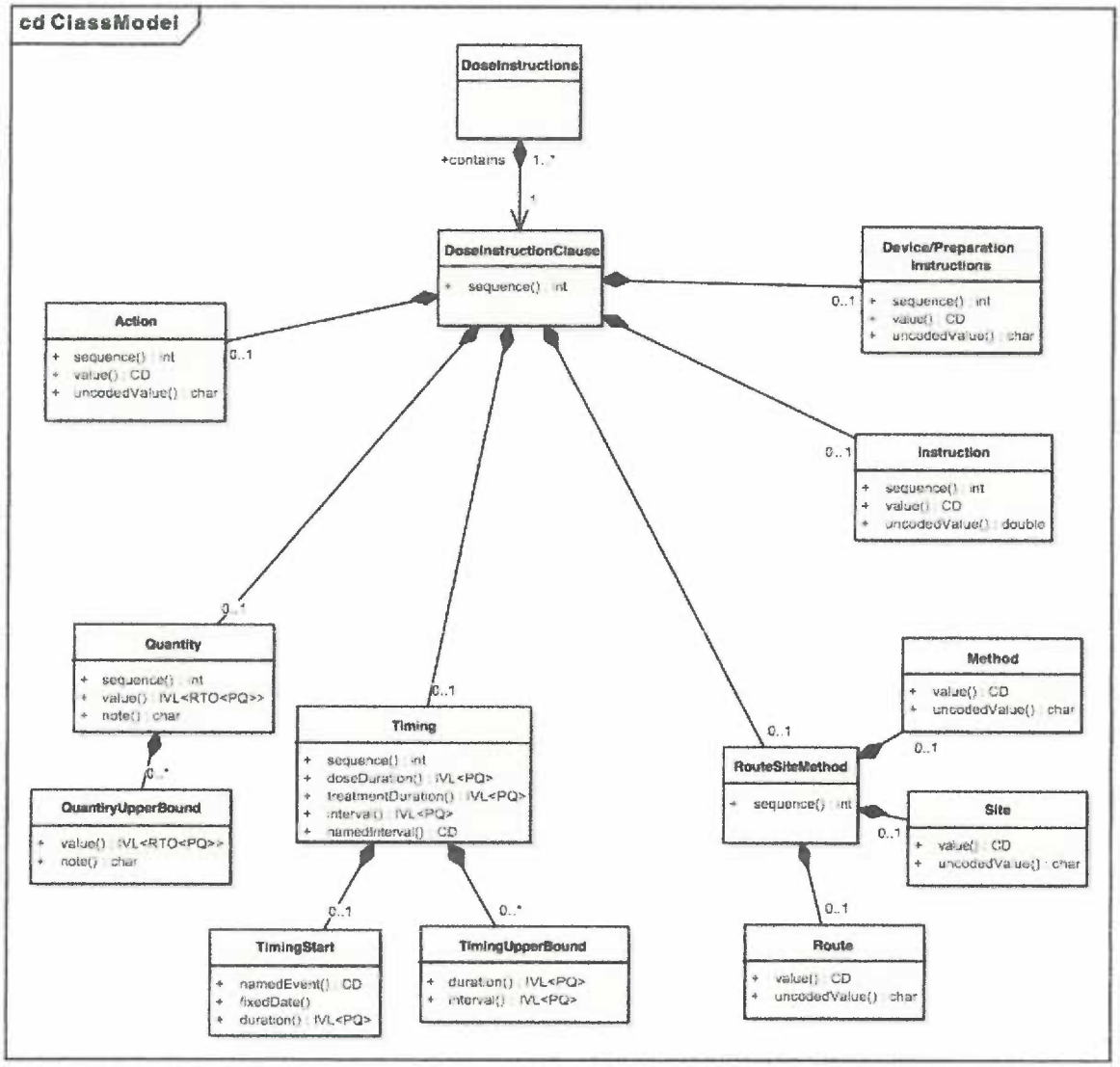
V. Conclusion

RxParse shows considerable promise as a general purpose text-mining tool for free-text *sig* data. The parser correctly identifies all 17 dose components nearly 70% of the time. Its overall recall of 87% and precision of over 99% are comparable to or better than published statistics for other prescription parsers. In addition, because RxParse uses lexical pattern matching with regular expressions, it can be used to quickly parse massive datasets such as those in an enterprise data warehouse. As investigators face the challenge of deriving knowledge from the data stored in modern CPOE systems, validated tools like RxParse may prove to be enormously useful.

The original project objectives were to 1) design and evaluate a free-text prescription parser that would allow researchers to calculate derived values such as total daily dose of a medication, 2) adhere to existing terminologies and data standards where appropriate, 3) use technologies that would allow reuse in other settings, and 4) validation of the tool against a representative clinical dataset prior to general use. This Capstone work has achieved these objectives and demonstrates that RxParse could be a valuable tool for researchers and informaticists alike.

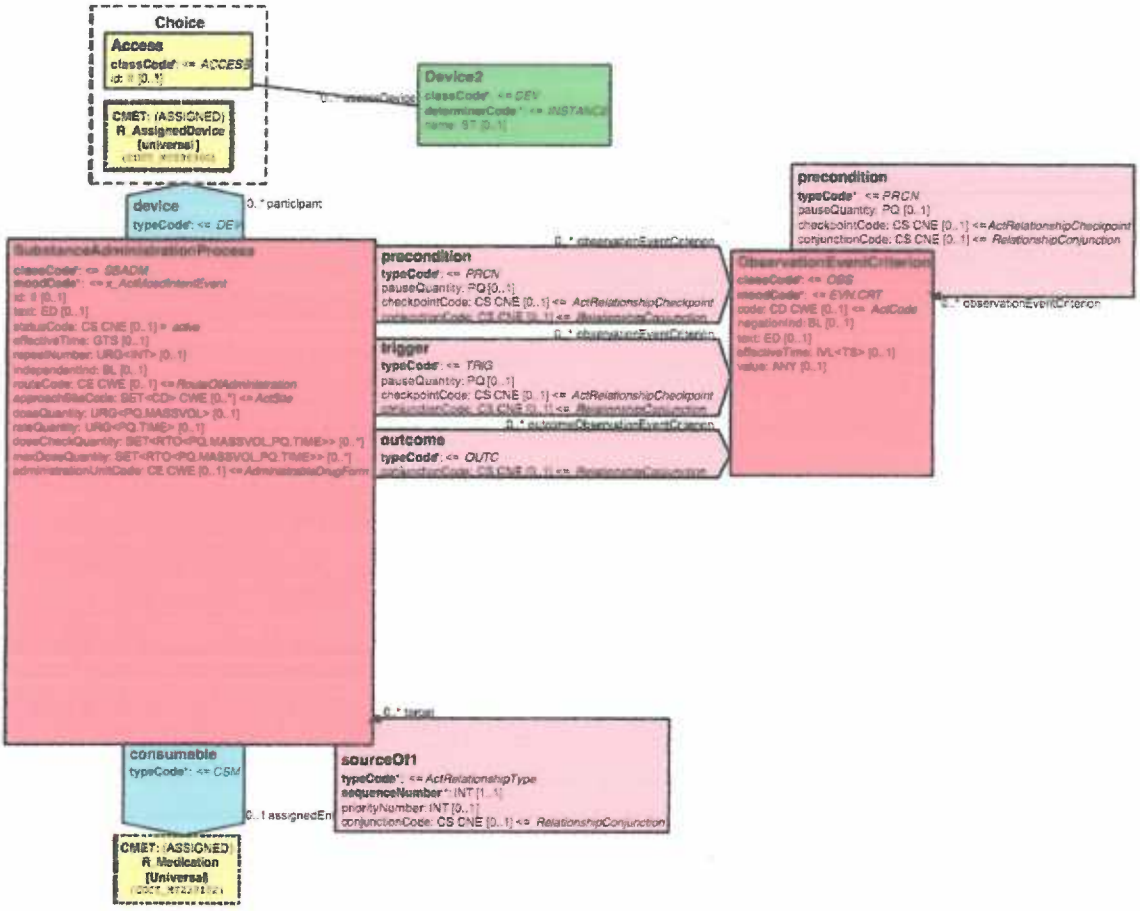
Appendix A - NHS Dose Syntax Model

The NHS Dose Syntax model allows for a variety of discrete elements, including the action, quantity, timing, route, site, and method of drug delivery.

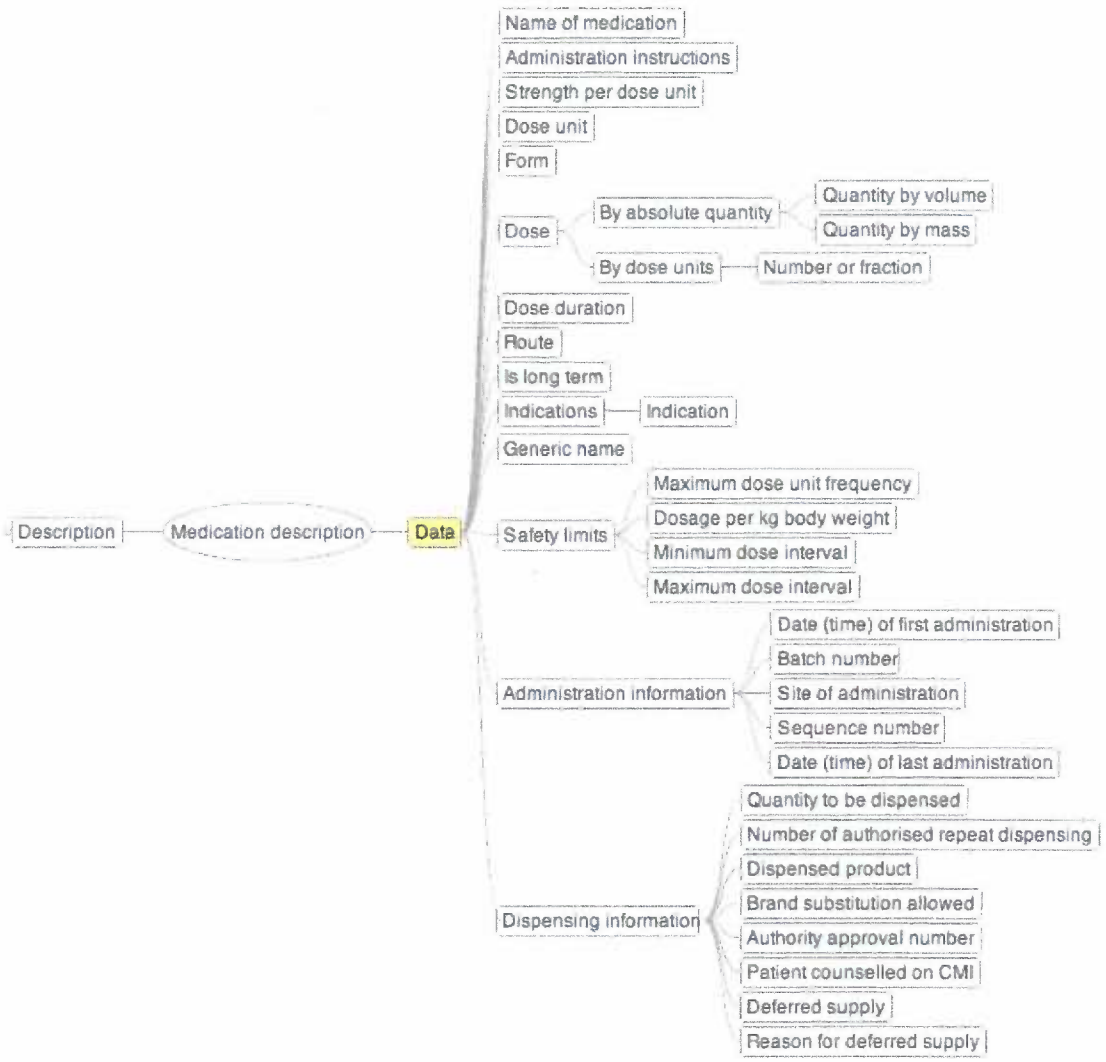


Appendix B - Dosage Instructions Representation in HL7 V3 Pharmacy Information Model

The HL7 v3 Pharmacy Information Model contains a subset of the elements modeled by the NHS Dose Syntax.



Appendix C - OpenEHR Medication Archetype



Appendix D - Capstone Prescription Data Model

Command	The command describes the imperative or directive that indicates how the medication should be given. This is modeled after the NHS Dose Syntax element called "Action" **Example: "take 2 tablets every 4 hours" – Command = "TAKE"**
Dose Low	If a single numeric dose is specified, this is the numeric value. **Example: "swallow 2 tablets" - DoseLow = 2** If a dose range is specified, this is the lower end of the dose range. **Example: "swallow 2-4 tablets" - DoseLow = 2**
Dose High	If a single numeric dose is specified, this value is null. If a dose range is specified, this is the upper end of the dose range. **Example: "swallow 2-4 tablets" - DoseHigh = 4**
Dose Unit of Measure	The unit of measure if explicitly stated. **Example: "swallow 2 tablets" - DoseUOM = tablets**
Alternate Dose	Some scripts have parenthetical alternate doses, which should be represented in this field. **Example: "take 5 ml (1 tsp) by mouth" AltDose = 1**
Alternate Dose Unit of Measure	Some scripts have parenthetical alternate units of measure, which should be represented here. **Example: "take 5 ml (1 tsp) by mouth" AltDoseUom = teaspoon**
Route	The route is the delivery mechanism for a medication. Examples include "oral", "intravenous", "topical", "rectal", and "inhaled" **Example: "Apply topically to rash". Route = "TOPICAL"**
Frequency Low	Frequency-based *sigs* describe the **number of doses per given unit of time**. If the dose is described with a frequency, the number of doses is represented here. If the dose has a frequency range, the lower bound is represented here. **Example: "1 tab 2-4 times per day". FreqLow = 2**
Frequency High	If the dose is described with a frequency range, the higher end of the range is represented here, otherwise null. **Example: "1 tab 2-4 times per day". FreqHigh = 4**
Frequency Unit of Time	If the dose is described with a frequency, the specified unit of time over which the dose is repeated is represented here. **Example: "1 tab 2-4 times per day". FreqUOT = day**
Interval Low	Interval-based *sigs* describe the **elapsed time between doses**. If the dose is described with an interval, the numeric interval is represented here. If a dose has an interval range, lower end of the range is specified here. **Example: "1 tab every 4-6 hours". IntLow = 4**
Interval High	If the dose is described with an interval range, the upper end of the numeric interval range is represented here, otherwise null. **Example: "1 tab every 4-6 hours". IntHigh = 6**
Interval Unit of Time	The unit of measure for the elapsed time between doses in interval-based sigs. If this is an interval-based *sig* and the unit of time is explicitly stated, the unit of time is represented here. **Example: "1 tab every 4-6 hours". IntUOT = HOURS**

Duration Low	If the sig is specified to be given for a certain duration, the numeric duration or lower end of a duration range is represented here. **Example: "by mouth for 3-6 weeks". DurationLow = 3**
Duration High	If the sig specifies a duration range, the upper value of the range is represented here, otherwise null. **Example: "by mouth for 3-6 weeks". DurationHigh = 6**
Duration Unit of Time	If the sig specifies a duration, the unit of time for which the duration applies, otherwise null. **Example: "by mouth for 2 days". DurationUOT= "days"**
Medication is PRN	Some medications are to be given conditionally or at the patient's discretion. Common descriptions of condition-based meds include "PRN" ("*pro re nata*" in Latin, or "as needed"). If the medication clearly states it is to be given only for certain conditions, this value is "True". **Example: "take as needed for fever". IsPRN = True.**

Appendix E - Word Frequency Analysis

Top 150 tokens found in a subset of 10,000 ambulatory scripts from CHOP:

Frequency	Token	Frequency	Token	Frequency	Token
5188	daily	251	area	71	puff
3897	by	245	pain	70	hrs
3883	mouth	227	week	68	qid
3548	for	202	morning	67	gtts
3205	days	199	am	66	night
2608	mg	199	bedtime	66	cap
2207	tsp	195	fever	65	flare
2045	twice	188	nostril	64	repeat
1838	as	187	ear	60	mouthpiece
1482	every	187	tabs	59	nss
1468	times	184	per	58	please
1326	to	184	take	57	if
1318	hours	181	areas	54	chewable
1112	tab	181	asthma	54	trainer
1045	needed	177	dispense	52	face
1034	ml	173	s	48	diaper
1027	day	172	teaspoon	47	qd
1001	x	170	capsule	45	into
942	and	161	weeks	45	small
875	in	155	tid	45	ten
840	via	147	tablets	44	patch
811	puffs	145	vial	44	wheezing
802	directed	142	prn	43	meals
775	on	134	dx	42	cc
755	apply	126	be	41	pm
690	two	124	inhaler	40	skin
680	tablet	119	give	40	prior
639	affected	117	spray	40	flares
577	or	117	four	39	both
568	po	116	used	38	6ml
540	a	114	of	36	scalp
535	spacer	114	medications	36	not
517	one	110	mask	35	w
507	dose	106	weekly	35	eczema
499	with	105	drop	34	pack
474	now	103	q	34	minutes
447	once	99	teaspoons	33	2x
431	cough	96	before	32	hour
425	three	93	evening	32	diagnosis
387	nebulizer	93	after	31	plan
372	then	91	until	31	half
362	use	89	medium	30	amount
339	wheeze	82	rash	30	today
315	bid	81	eyes	30	4ml
314	neb	80	congestion	30	im
306	drops	80	5ml	29	syringe
295	at	79	up	29	water
294	the	79	capsules	29	continue
291	each	77	sprays	29	itching
259	eye	74	saline	28	capful

References

[1] Barnett GO, Jenders RA, Chueh HC. The computer-based clinical record – where do we stand? Ann Intern Med. 1993;119(10):1036-41.

[2] United States. General Accounting Office. Automated Medical Records: Leadership Needed to Expedite Standards Development: report to the Chairman/Committee on Governmental Affairs, U.S. Senate. Washington, D.C.: USGAO/IMTEC-93-17; April 1993.

[3] RxHUB National Patient Health Information Network. Rx Hub – Home. Available at: http://www.rxhub.net/index.php. Accessed: May 4, 2008.

[4] NHS Dictionary of Medicine + Devices. DOSE Syntax. Available at: http://www.dmd.nhs.uk/dossyntax.html. Accessed: May 4, 2008.

[5] The openEHR Foundation. Home Page. Available at: http://www.openehr.org/home.html. Accessed on: May 4, 2008.

[6] NCPDP. National Council for Prescription Drug Programs. Available at: http://www.ncpdp.org/. Accessed: May 4, 2008.

[7] National Committee on Vital and Health Statistics. Agenda of the December 7-8, 2005 NCVHS Subcommittee on Standards and Security Hearings. Available at: http://www.ncvhs.hhs.gov/051207ag.htm. Accessed: May 4, 2008.

[8] Health Level Seven. Home Page. Available at: http://www.hl7.org/. Accessed: May 14, 2008.

[9] Blue Wave Informatics, LLC. Dose Syntax Specification. Available at: http://www.bluewaveinformatics.co.uk/BW/3_2/DocumentList.htm. Accessed: May 4, 2008.

[10] Blue Wave Informatics, LLC. Application Links. Available at: http://www.bluewaveinformatics.co.uk/BW/3_2/application_links.htm. Accessed: May 4, 2008.

[11] Leslie H. International developments in openEHR archetypes and templates. HIM J. 2008;37(1):38-9.

[12] Wikipedia, The Free Encyclopedia. Regular expression. Available at: http://en.wikipedia.org/wiki/Regular_expression. Accessed on: May 4, 2008.

[13] Friedl JEF. Mastering Regular Expressions. 2nd ed. Sebastopol, CA: O'Reilly and Associates; 2002.

[14] Turchin A, Pendergrass ML, Kohane IS. DITTO - a tool for identification of patient cohorts from the text of physician notes in the electronic medical record. AMIA Annu Symp Proc. 2005;:744-8.

[15] Turchin A, Kolatkar NS, Grant RW, Makhni EC, Pendergrass ML, Einbinder JS. Using regular expressions to abstract blood pressure and treatment intensification information from the text of physician notes. J Am Med Inform Assoc. 2006 Nov-Dec;13(6):691-5. Epub 2006 Aug 23.

[16] Teich JM, Hurley JF, Beckley RF, Aranow M. Design of an easy-to-use physician order entry system with support for nursing and ancillary departments. Proc Annu Symp Comput Appl Med Care. 1992;:99-103.

[17] Kraus S, Blake C, West SL. Information Extraction from Medical Notes. In Kuhn KA, Warren JR, Leong TY, editors. Proceedings of the 12th World Congress on Health Informatics– Building Sustainable Health Systems (MedInfo). Brisbane, Australia; 2007. p1662-4.

[18] Meystre SM, Haug PJ. Comparing natural language processing tools to extract medical problems from narrative text. AMIA Annu Symp Proc. 2005;525-529.

[19] Wilke RA, Berg RL, Peissig P, Kitchner T, Sijercic B, McCarty CA, McCarty DJ. Use of an electronic medical record for the identification of research subjects with diabetes mellitus. Clin Med Res. 2007;5(1):1-7.

[20] Barrows Jr RC, Busuioc M, Friedman C. Limited parsing of notational text visit notes: ad-hoc vs. NLP approaches. AMIA Annu Symp Proc. 2000;51-5.

[21] Viera AJ, Garrett JM. Understanding interobserver agreement: the kappa statistic. Fam Med. 2005;37(5):360-3.

[22] Landis JR, Koch GG. The measurement of observer agreement for categorical data. Biometrics. 1977;33:159-74.

[23] VassarStats. Clinical Calculator. Available at: http://faculty.vassar.edu/lowry/clin1.html. Accessed on: May 5, 2008.

[24] CPAN. Time::HiRes. Available at: http://search.cpan.org/dist/Time-HiRes/HiRes.pm. Accessed on: May 5, 2008.