Extracting structure from human-readable semistructured text

Elaine Angelino

ABSTRACT

The explosion of Web data over the past fifteen years has fostered a rich body of research in extracting structure from semistructured HTML and XML documents. Today, we are also in the middle of an explosion of semistructured documents that originate from *outside* the Web domain. This latter kind of semistructured data is everywhere: in electronic medical records (EMRs), government reports, digital humanities archives, and datasets from many other domains. In contrast to HTML or XML, *semistructured text* that has not been marked up is typically *human-readable* (HR), and its structure implicitly reflects a schema. Our high-level goals are to (1) explicitly recover meaningful structure in semistructured text corpora, and (2) demonstrate that this effort enables more accurate analytics and facilitates or enriches other applications. To motivate our approach and work, we highlight the differences between HR and marked-up semistructured text and illustrate concrete use cases of HR semistructured text. We identify specific instances of implicit structure along a spectrum of "structuredness" commonly found in HR semistructured text and describe a corresponding array of methods for extracting structured features. It has not been our goal to develop "the best" such methods, but to instead present a principled framework for evaluating and combining multiple extraction methods in the context of specific data analytic tasks. We also present example applications that can be quickly built on top of a substrate of features extracted from semistructured text. For concreteness, we focus this report around semistructured text found in EMRs and evaluate our framework using a large corpus of text reports from real EMRs exhibiting highly heterogeneous schema.

1. INTRODUCTION

An incredible amount of digital data exists in our world today. On one extreme, there continues to be an ever-growing number of traditional relational databases, which use schema to explicitly structure data. On the other extreme, there is a gigantic corpus of unstructured media content – including digital books, images and videos – sometimes with, but often without any descriptive metadata. While we have developed many techniques for analyzing and interacting with corpora of unstructured data, these techniques inherently rely on identifying and utilizing structure within and relating data items. For example, a basic search engine for images may associate with each image descriptive keywords inferred from related text content. We can also associate with each image a set of extracted features and computed statistics. So, by determining the most dominant color in each image, we could support an interface allowing the user to filter images by color, in addition to subject matter.

Human effort is typically required to specify useful schema for structuring data, and this effort allows us to more expressively interact with data. When we know a dataset's schema, we can perform sophisticated queries and augment data exploration and search. We can further leverage schema structure in our data analysis, notably, when we use this structure to extract features for input to statistical and machine learning techniques. Finally, semantically meaningful schema aid human interpretation of data, as well as schema integration of multiple datasets.

In between the two extremes of structured and unstructured data, there exists a wealth of semistructured data. Buneman characterizes semistructured data as *implicitly* containing "the information that is normally associated with a schema ... which is sometimes called *self-describing*." He mentions three motivating use cases for semistructured data: performing database operations on data like the Web, sharing data between different databases and browsing structured data [Buneman 1997]. If we can discover underlying structure of semistructured data, e.g., a schema, we can then apply structured data management and data analysis techniques to semistructured data.

We restrict our attention to *human-readable* (HR) semistructured text, which is abundant, arises across many domains, exhibits a wide range of 'structuredness,' and allows us to exploit existing techniques from natural language processing. At this point, let us clarify what we mean by semistructured text (§1.1) and HR semistructured text (§1.2). We then provide concrete examples of HR semistructured text (§1.3) to motivate our approach in extracting structure from such text (§1.4).

1.1 Semistructured text

We consider semistructured text to be any text ranging from the contents of relational databases to natural language prose. In fact, many relational databases contain semistructured text stored as blobs, as we shall see in our examples from electronic medical records (EMRs) [and other domains]. We want to discover schema that can be used to meaningfully structure corpora of semistructured text,[1] for ingest into relational databases or input into data analysis algorithms. We assume that tokens, rather than characters, for some reasonable tokenization, are the basic units of any text.

HTML and XML web pages are examples of semistructured text. Many web pages are produced by filling in an HTML template with queries to a database; product pages, such as those at `www.amazon.com`, are generated in this fashion.

1.2 Human-readable (HR) semistructured text

HTML and XML use tag elements to explicitly encode structure. Plain text often uses "visual tags" to implicitly communicate structure, e.g., visual cues such as new lines and white spaces. Humans use visual cues to convey structure when manually producing text, such as when writing a recipe, formatting a brochure or typing up lecture notes. In general, a single semistructured text document contains a hodgepodge of different kinds of structured regions, such as tables and lists embedded between paragraphs of text.

Semistructured text arises when data *intended for human consumption* is stored digitally without regard to the fact that different representations enable different kinds of human interaction. While semistructured text is intuitive for humans to produce and consume, people – who may or may not have produced the text – often want to interact with the text in a structured way. For example, consider a string of text representing a table with two named columns of numerical data. When a human looks at this text in a visual environment such as a text editor, she can immediately identify the two columns and separate the column names from the data based on visual cues. However, if she wants to easily produce a scatter plot of one column against the other, she must make this structure explicit, e.g., by importing the data into spreadsheet software.

Semistructured text arises in many domains, notably those undergoing transitions to ubiquitous digital records and archives, including medicine, government and the humanities. Congressional incentives for hospitals to use EMRs by 2015 [hit 2009], the Open Government Initiative, which aims to increase transparency in part through publishing digital collections of government data on the web [2] and the creation of the Office of Digital Humanities at the National Endowment for the Humanities are all indicators of this trend.

1.3 Examples of HR semistructured text in EMRs

Because EMR systems must manage complex data associated with diverse and heterogeneous schema, it has been challenging to design a system that integrates all the data while explicitly representing the schematic structure. Furthermore, EMR systems must be structured to be easily extensible as new medical procedures and technologies introduce new data management requirements. The urgent need for digitization of health records has led to the development of EMR systems that are operating in real hospitals and useful to real health care professionals. Since such an EMR system is driven by immediate healthcare needs, its design is not motivated by new applications of EMR data. We have seen these tensions reflected in the data management design decisions of current EMR systems, and in particular their choice of lightweight schema.

While EMR systems are managed by databases, they are often archives of HR semi-structured text documents, each with a limited amount of explicitly structured metadata. This "document oriented" approach is described by a simple relational schema: a single logical table containing one field for the EMR documents and one field for each piece of per-document metadata.[3] However, EMR systems manage data that is heterogeneous with respect to content and use, and, as a consequence, structure and format. In addition to basic patient information, such as demographic and biometric data, a patient's records include diverse information such as quantitative laboratory test results (e.g., blood tests), specialty-specific reports (e.g., for X-rays and electrocardiograms), and lengthy notes (e.g., written from a specialist to a primary care physician). Each kind of report can be described by and is often generated according to its own special schema.

[1] "One-shot" schema discovery for individual documents is beyond the scope of our report.

[2] `www.whitehouse.gov/open`, `www.data.gov`

[3] The schema usually includes a small number of additional tables, e.g., separate tables for patients and providers.

From our own experience with EMR systems as well as interviews we conducted with several health informatics experts we found that many EMR systems do not explicitly represent these report-level schema. Instead, each report is represented within the EMR system as a string of HR semistructured text, plus a few key pieces structured of metadata, such as the date and a unique identifier for the patient. The schema of each report type is *implicitly and visually* represented in the report text.

Figure 1 shows electrocardiogram (ECG) reports from the Partners HealthCare Longitudinal Medical Record (LMR) system. We show two examples for comparison. All examples of EMR entries in this report have been manually anonymized. These reports contains several kinds of structured information:

—*Key-value pairs*, e.g., `Report Number: 0000000W`, where `Report Number` and `0000000W` are the key and value, separated by a colon and single white space. Multiple key-value pairs may occur in the same line of text, as in the first line, but are not explicitly separated by an unambiguous delimiter. Also, notice that values are in general tuples of data values, e.g., the data value of `Electrocardiogram Report # 00-00000W 01/01/01 14:02` can be represented as the 3-tuple, `(00-00000W, 01/01/01, 14:02)` The key provides *metadata* necessary for interpreting the corresponding *data* value; we discuss key-value pairs futher in §3.3.

—*Numerical data values* corresponding to medical data measurements, e.g., `VENT. RATE 99 BPM`. Numerical data values are really a special kind of key-value pairs whose values must be recognized as numbers. As with key-value pairs, the data values are in general tuples, e.g., `P-R-T AXES 66 -10 95` represents a 3-tuple of integers. These numerical values require units for meaningful interpretation and the units are part of the metadata. Many quantitative measurements are examples of key-value pairs where the key and value do not correspond neatly to the left and right sides of formatted text but rather where the data value is surrounded by contextual metadata. We study quantitative data values in §3.4.

—*Verbose text* that provides interpretive, diagnostic, prescriptive and other further information. This text may be relatively *unstructured* as in a doctor's note or highly structured as with machine-generated output; we introduce the term *structured prose* to describe the latter. We study these kinds of structure in §3.5-3.6.

The two reports in Figure 1 are clearly similar, sharing many of the same data items and formatting structure for key-value pairs, numerical data measurements and the statement, `NORMAL SINUS RHYTHM`. We can also notice several structural differences:

—The first report is missing the value for `QT/QTC`, which can indicate that the EKG machine was unable to extract this measurement from the raw ECG signal.

—In the first report, the `P-R-T AXES` measurement includes three numerical values but the second report only has two. The latter is incorrect and is an example of human data entry error.

—The second report is longer than the first, containing more statements interpreting the EKG results, including statements comparing the two reports.

Even EMR systems that try to impose structure on text reports fail to expose their data in a way that would be automatically extractable. Figure 2 shows an example of a de-identified EMR entry from the Regenstrief Medical Record System / Indiana Network for Patient Care (RMRS/INPC) EMR system; it is also an ECG report. The text block on the left is a visual representation of the report. In contrast to the Partners EMR system, this text report is represented by XML rather than plain text. Unfortunately, the XML does not does not add any meaningful structure. We show this XML directly next to the visual representation, on the right.

For clarity, we have formatted the original XML by inserting line breaks so that it is easier to visually compare with the corresponding text representation. The XML schema in this example is representative of all EMR entries in this system: it contains a sequence of `<text>` tags, each containing zero or one `<title>` tags followed by at least one `<p>` tag. This schema describes a sequence of key-value pairs, where keys and values correspond to the contents of the `<title>` and `<p>` tags, respectively. However, closer inspection reveals that the XML does little more than capture the visual schema of the text. For example, the corresponding XML for lines containing two key-value pairs only explicitly represent the first key in a `<title>` tag, and concatenate the first value with the entire second key-value pair in the corresponding `<p>` tag. We can deduce that the XML schema derives from a naive attempt to infer the structure of an original plain text representation.

```
Report Number: 0000000W Report Status: FINAL
Type: EKG
Date: 01/01/01 14:02
Electrocardiogram Report # 00-00000W 01/01/01 14:02
VENT. RATE 99 BPM
PR INTERVAL 188 ms
QRS DURATION 100 ms
P-R-T AXES 66 -10 95
NORMAL SINUS RHYTHM
LEFT VENTRICULAR HYPERTROPHY WITH REPOLARIZATION ABNORMALITY
REFERRED BY: AARDVARK, M.D.,ALICE. REVIEWED BY: BEAGLE, M.D.,BOB
[report_end]
```

```
Report Number: 0000000W Report Status: FINAL
Type: EKG
Date: 02/02/02 13:25
Electrocardiogram Report for Accession # 00-00000W 02/02/02 13:25
VENT. RATE 76 BPM
PR INTERVAL 154 ms
QRS DURATION 88 ms
QT/QTC 422 474 ms
P-R-T AXES 55 94
NORMAL SINUS RHYTHM
NONSPECIFIC T WAVE ABNORMALITY
PROLONGED QT INTERVAL OR T-U FUSION, CONSIDER MYOCARDIAL DISEASE,
ELECTROLYTE IMBALANCE, OR DRUG EFFECTS
ABNORMAL ECG
WHEN COMPARED WITH ECG OF 01-JAN-2001 14:02,
T WAVE INVERSION NOW EVIDENT IN LATERAL LEADS
REFERRED BY: AARDVARK, M.D.,ALICE. REVIEWED BY: BEAGLE, M.D.,BOB
[report_end]
```

Fig. 1. Two ECG reports from the Partners HealthCare LMR system. They contain several kinds of structured information and are similar but have some structural differences.

```
Stationary ECG Study: XXXX University XXXX  XXXX XXXX
Test Date: XXXX
XXXX Name: XXXX
Patient XXXX: XXXX                  Room:
Gender: Male                      Technician:   JG
DOB: XXXX
XXXX Number: XXXX                 XXXX MD:   XXXX XXXX
Intervals                              Axis
XXXX: 46                       P:            32
PR: 132                      QRS:          -19
QRSD: 112                      T:            -33
QT: 472
QTc: 448
Interpretive Statements
Sinus bradycardia
T XXXX inversions in leads #, aVF are not XXXX, but
are more prominent compared to prior studies
Subtle ST-T changes elsewhere are nonspecific
Low QRS voltages in precordial leads
Abnormal ECG
Electronically Signed On XXXX XXXX by XXXX XXXX
```

```
<text_report>
<text><title></title><p>Stationary ECG Study<br/>XXXX University XXXX<br/>XXXX XXXX</p></text>
<text><title>Test Date</title><p>XXXX<br/></p></text>
<text><title>XXXX Name</title><p>XXXX<br/></p></text>
<text><title>Patient XXXX</title><p>XXXX                  Room:<br/></p></text>
<text><title>Gender</title><p>Male                      Technician:   JG<br/></p></text>
<text><title>DOB</title><p>XXXX<br/></p></text>
<text><title>XXXX Number</title><p>XXXX                 XXXX MD:   XXXX XXXX</p>
<p>Intervals                              Axis<br/></p></text>
<text><title>XXXX</title><p>46                       P:            32<br/></p></text>
<text><title>PR</title><p>132                      QRS:          -19<br/></p></text>
<text><title>QRSD</title><p>112                      T:            -33<br/></p></text>
<text><title>QT</title><p>472<br/></p></text>
<text><title>QTc</title><p>448</p>
<p>Interpretive Statements<br/>
Sinus bradycardia<br/>T XXXX inversions in leads #, aVF are not XXXX, but are more prominent<br/>
compared to prior studies<br/>Subtle ST-T changes elsewhere are nonspecific<br/>
Low QRS voltages in precordial leads<br/>
Abnormal ECG</p>
<p>Electronically Signed On XXXX XXXX by XXXX XXXX</p></text>
</text></text_report>
```

Fig. 2. Visual rendering and XML representation of an ECG report from the RMRS/INPC EMR system. The XML does not add meaningful structure beyond visual formatting.

1.4 The problem

Existing methods for extracting information from semistructured data tend to focus on self-describing data, where the *metadata* is explicitly distinct from the corresponding *data values*. HTML and XML are clear examples of self-describing semistructured text: they utilize tag elements to convey semantic meaning. These tags are distinct from the actual content, and thus distinguishable as metadata. Unfortunately, the implicit structure of HR semistructured text is not self-describing. **Our primary challenge in extracting structure from semistructured text is to separate data values from metadata**.

To a human, semistructured documents, such as EMRs, contain a lot of inherent structure. To a computer, these semistructured documents are simply blobs of text, without any explicit structure beyond the fact that they are strings of characters. When presented with a corpus of semistructured documents, human experts can manually determine their schema, but this process is labor-intensive and can be difficult for large, heterogeneous datasets whose schema cannot be inferred from a single or small number of data examples.

When it comes to structuring semistructured documents, there is currently no tool that "just does the right thing" for arbitrary datasets. There is a tremendous amount of interest in this area, and recent research has been particularly directed toward reverse-engineering the schema of HTML and XML documents (§7). To our knowledge, less research has studied this problem for HR semistructured text. Our long-term goal is to design a system that, when presented with a corpus of semistructured text documents, *automatically* discovers the schema and then populates a database by transforming the input documents into records conforming to the discovered schema. This structured database would be a powerful back-end capable of supporting many applications, including rich queries and search, data mining algorithms, and data visualizations for human interaction and exploration.

As a concrete and important case study, we study semistructured text found in electronic medical records. In the next sections, we motivate our work by presenting use cases for *structured* EMR data (§2), describe different kinds of structure found in HR semistructured text (§3) and methods for extracting structured features from this text (§4). Using a large corpus of text reports from real EMRs exhibiting highly heterogeneous schema, we next evaluate the impact of different methods for extracting structured features in a specific data analytic setting (§5) and present examples of applications built on top of structured EMR text semistructured text (§6). Finally, we wrap up with a brief discussion of our work including some future directions (§8).

2. USE CASES FOR STRUCTURED EMR DATA

EMR systems are designed for human consumption. Thus, the fact that EMR data is commonly stored as HR semistructured text does not present an immediate problem for the health care professionals who interact with these systems to directly care for patients. However, there is powerful information "hidden" within and across EMR data that we can exploit if we can explicitly represent their structure. In this section, we present a series of potential use cases of EMR data. For each use case, we describe required structured data and challenges of extracting this data from the raw HR semistructured text commonly used to represent EMR data.

2.1 Identifying and understanding health trends and statistics

Scenario: Government agencies such as the Centers for Disease Control and Prevention and other policy makers, hospitals, insurance and pharmaceutical companies, epidemiologists and public health researchers are all agents interested in understanding temporal, spatial trends and aggregate statistics characterizing health care and disease. Specific goals include detecting epidemics and identifying their origins in real-time, monitoring the spread and virulence of different influenza strains, analyzing regional differences in health care, tracking the adoption of new medical technologies and drugs, identifying correlations between symptoms and diseases, discovering unknown side effects of new drugs, and assessing the overall health of groups of patients. EMRs contain rich information about populations of patients over time that these organizations and researchers can leverage to study population-level trends.

For example, a visualization in the spirit of Google's Ngram Viewer [ngr 2010] would help researchers quickly generate hypotheses about health trends and correlations by plotting frequencies of n-grams in a corpus of EMRs as a function of time. A researcher might use such a tool to ask, "Is the frequency of the phrase *cardiac disease* increasing, decreasing or stable?", "How large are seasonal fluctuations of *Allegra*, a popular allergy drug?" and "Do there appear to be long-term trends in the relationship between *breast*

cancer and *ovarian cancer*?" Finer levels of data aggregation such as at the granularity of single patients or even individual patient event entries, would enable researchers to ask more sophisticated questions, e.g., by analyzing the co-occurrences of sets of features extracted from EMRs. This would support questions such as, "Is the use of the antibiotics *vancomycin* and *doxycycline* correlated or anticorrelated?"

Google Flu Trends is a related tool that "uses aggregated Google search data to estimate flu activity" by leveraging a set of search terms discovered to be "good indicators"[4].

Data challenge: It is straightforward and useful to extract n-grams from a corpus of EMRs (§5). We follow techniques similar to those behind Google's Ngram Viewer, which extracts n-grams from digitized books aggregated by year [cite Nature paper]. Thus, we perform only minimal data processing, using standard natural language processing techniques such as stemming and stop word removal.

N-gram extraction is an unsupervised and fast domain-agnostic technique for extracting features from text. Other purely statistical techniques that capture other structure include *template extraction* to identify constant versus varying regions across multiple documents generated according to the same template [cite Web template] and *chunking*, i.e., light sentence parsing, to extract sequences of words likely to be meaningful phrases [cite Tango].

However, n-gram extraction is naive in the sense that does not explicitly perform many functions required for understanding text, including:

—parsing sentences for meaning (the phrase *lung and prostate cancer* refers to two different diseases, *lung cancer* and *prostate cancer*),

—associating semantic meaning with n-grams (*lung cancer* and *prostate cancer* are both kinds of *cancer*),

—recognizing synonyms (*heart attack*, *myocardial infarction* and *myocardial infarct* are all common synonyms of *myocardial infarction* found in EMRs) and

—performing sentiment analysis (*no evidence of myocardial infarction* has the opposite meaning of *acute myocardial infarction is present*).

In order to take a semantically-driven, natural language based approach, we can exploit existing software tools. For example, the National Library of Medicine maintains MetaMap[5], a software stack that parses medical text into grammatical phrases using NLP techniques and maps these phrases onto standardized concepts within a semantic tree using the Unified Medical Language System (UMLS)[6]. We use MetaMap to extract explicit medical concepts from EMRs (§5).

There are some challenges to using MetaMap. It is a complex tool that has been under development for over a decade and is built on top of large and manually curated databases of domain-specific knowledge including dictionaries and thesauruses that must be updated, e.g., to add new medical techniques and drugs. It also requires a significant amount of time to process large amounts of data. Due to patient data privacy concerns, data processing must occur in a secure environment that is often computationally constrained, e.g., a particular computer with limited capacity for parallelization. In such a setting, it may be impractical to utilize MetaMap in real-time to process a live and growing EMR system.

2.2 Real-time in-hospital data monitoring

Scenario: Individual hospitals can leverage real-time monitoring of internal EMR data for diverse applications. For example, an EMR system should enforce quality controls that instantly detect anomalous data entry due to human error or machine malfunction. Instead of having to specify acceptable data values and terminology upfront, the system could take a Google-style "Did you mean ...?" approach that uses statistics of observed data values, spellings, etc. to create prior expectations for correct data entry. As another example, an EMR system should be able to detect unusual events such as clusters of methicillin-resistant *Staphylococcus aureus* (MRSA) infections among patients suggesting a potential outbreak within the hospital. A visualization of "trending topics" within a hospital or particular department could summarize current and recent patient demographics and administered treatments or procedures and highlight unusual events.

[4]http://www.google.org/flutrends/
[5]http://metamap.nlm.nih.gov/
[6]http://www.nlm.nih.gov/research/umls/

Data challenge: In addition to requiring the kinds of language-oriented feature extraction described previously (§2.1) – e.g., n-grams or medical concepts – some applications may require extraction of quantitative data. For example, an EMR quality control system that learns acceptable ranges of measured data values for each instrument must identify numerical data values within the EMRs as well as contextual information for interpreting the numbers: the name of the measurement, units, etc. We describe techniques for extracting quantitative data and relevant metadata, as well as example applications (§5).

2.3 Studying the epidemiology of patient populations

Scenario: A team of epidemiologists is studying the effectiveness of a new cholesterol-lowering drug, `Drug X`, that was recently approved by the Food and Drug Administration. They are studying a large and diverse cohort of cardiac patients, all of whom have been prescribed either `Drug X` or the older cholesterol-lowering `Drug Y`. Because the researchers have access to the patients' EMRs, they have the opportunity to perform a "retrospective" study evaluating the effectiveness of `Drug X`. While `Drug X` has already been studied in several randomized controlled trials, it has never been studied in 'wild' populations of cardiac patients. The total 'wild' patient population taking `Drug X` is far greater in number than any of the clinical trial populations, and so epidemiologists might be able to learn more about `Drug X`'s effectiveness, rare side effects, long-term effects, etc., potentially with higher statistical significance than was established by the controlled trials.

However, the "wild" population is far more heterogeneous in terms of demographics and clinical history than any of the trial populations; all this variation between patients confounds the association between patients taking `Drug X` and measurable outcomes. For example, doctors may tend to prescribe `Drug Y` to older patients because it has less severe side effects than `Drug X`. At the same time, older patients tend to experience fewer benefits from both drugs compared to younger patients. Patient age thus confounds the perceived effectiveness of `Drug X` versus `Drug Y`. In order to fairly compare the group of patients taking `Drug X` to the others taking `Drug Y`, the epidemiologists need to adjust for the fact that the ages of patients in the two groups are not evenly matched.

Data challenge: The epidemiologists already utilize easily accessible and limited structured information contained in the EMRs and medical claims reports, such as basic demographic information and standard diagnostic codes. While this data accounts for some of the confounding differences between patients, EMRs contain rich semistructured information that the researchers suspect will help identify additional confounding factors. We demonstrate that this is indeed the case and show that the semistructured text reports can reveal significant and new confounders, thus augmenting the set of previously known confounders from the structured data (§5). We identify potential confounders by extracting domain-agnostic features, such as n-grams, and show that these also capture confounding factors. We are also in the process of extracting medical concepts corresponding to events, tests, diagnoses, etc. and standard medical measurements such as blood pressure and cholesterol.

2.4 Understanding a patient's medical history

Scenario: A doctor meets a new patient and needs to obtain a wholistic picture of her current health by understanding her medical history. Many scenarios can lead a patient to new doctors, e.g., when she moves to a new area, transitions from child to adult health care or develops other specialized needs. It is a standard and valuable practice for health care professionals to interview patients, but interviews are not always possible and when they are, the information obtained is likely biased, incomplete and inaccurate.

Data extracted from a patient's EMRs could be displayed in an interface that summarizes the patient's history to enable a doctor to rapidly construct a complete picture of the patient. Such an interface must convey important information more efficiently and simply than direct browsing of individual EMR entries and must provide information that supplements a patient interview. For example, this interface could present a timeline of medical events, grouping together recent metrics corresponding to "normal health" and separately highlighting anomalous or non-standard laboratory results, symptoms and complaints leading to these tests, interpretations of these results, etc.

Data challenge: A patient's EMRs contain rich information but are complex and difficult to quickly read and interpret. These can easily contain hundreds or thousands of separate entries spanning multiple decades

and include highly heterogeneous data including standard laboratory reports, accounts of emergency events and letters between specialists and primary care physicians. Not all of this data is relevant to understanding a patient's overall health. Creating this interface requires identifying and aggregating over key events and pieces of data buried within a patient's EMRs, including the kinds of language-oriented and quantitative information described previously (§2.1-§2.2). This interface likely also requires higher-level semantic interpretation of extracted data in order to highlight and group together important items such as diagnoses and treatments. We can obtain these kind of semantics from a tool such as MetaMap, as described earlier (§2.1).

2.5 Giving patients access to EMRs

Scenario: The rising prevalence of EMRs has prompted interest in giving patients more direct access to their own medical data. What are useful and accessible ways for patients and families to interact with their own medical data? For example, two parents whose children were both diagnosed with the same condition may want to compare their experiences with different doctors. Like the health care professionals who must interpret a patient's EMRs, the patient could also benefit from interacting with a summary view of his EMRs. An EMR interface designed for patient consumption will be different from one for doctors, since patients have different needs and medical expertise. Patients likely benefit from minimal use of medical jargon, explanations of test results, information about prescriptions, etc.

Data challenge: Here, the challenge is to start with raw EMR data – generated by and for health care workers – and transform it into information supporting a patient-facing EMR interface. This requires extracting medical concepts and translating them into terms better understood by laypeople, identifying test results and their interpretations, identifying prescribed drugs and mapping them to a database of drug facts, etc.

3. STRUCTURE IN HR SEMISTRUCTURED TEXT

In this section, we describe structure at two levels of granularity: (1) *document-level* structure characterized by objects such as key-value pairs and paragraphs of unstructured text (§3.1) and (2) the lower-level *substructure* of each of these objects (§3.3-3.6). *Key-value pairs* lay the foundation for how we understand several kinds of substructure commonly found in HR semistructured text (§3.2). We identify different substructures including three special cases of key-value pairs: *formatted key-value pairs* (§3.3), *quantitative data* (§3.4) and *structured prose* (§3.6). For these substructures, we view the text as containing both data values and metadata and so our challenges are to both distinguish between the two and determine what metadata goes with what data. We also discuss a range of standard substructures in *unstructured text* from the field of natural language processing (NLP) (§3.5). These structures inform what features we extract as well as our extraction methods (§4).

3.1 Document-level structure

HR semistructured text documents exhibit high-level structure: a single document uses visual formatting to (1) organize the text into meaningful regions and (2) convey relationships between these regions. For example, a layperson looking at one of the sample ECGs in §1.3 can use visual cues to write down a simple schema that captures and distinguishes between the regions of key-value pairs and less structured text. Note that it is common for document-level structure to be hierarchical; e.g., this report is hierarchical because there are high-level sections containing subsections. Finally, document-level structure can come from a variety of sources. For example, a person writing a report may rely on intuition and heuristics to figure out where to insert section headings and line breaks; in contrast, the blood test reports in an EMR system might all be automatically generated according to a common template.

3.2 Key-value pairs

A *key-value pair* associates a data value with a named attribute, or key, used to identify the value. We take the straightforward view that the structure of a key-value pair is revealed when it is expressed as a tuple, *(key, value)*.

3.3 Formatted key-value pairs

A *formatted key-value pair* is a HR semistructured text representation of a key-value pair that uses a visual cue to separate the key from the value. For example, we can infer that ‘‘`Gender: Male`’’ uses a colon plus a single whitespace to to separate the key `Gender` from the value `Male`. In HR semistructured text, the key tends to convey semantic information about how to interpret the corresponding value, and so we view the key as providing *metadata* associated with the *data* value.

3.4 Quantitative data

In HR semistructured text, quantitative data is often represented as a key-value pair, i.e., the numerical data value itself is accompanied by a named attribute. For example, ‘‘`TEMP 98.7C`’’ corresponds to a measurement and can be thought of as a string with three parts: (i) the measurement name, (ii) the data value and (iii) measurement units. As a key-value pair, we consider parts (i) and (iii) to correspond to the key and part (ii) to the value. The numerical data value (ii) is easily distinguishable from the metadata, e.g., if we assume that it represents an integer, decimal or simple expression such as a fraction of integers.

EMR entries exemplify how rich numerical data can be buried within HR semistructured text: specialized reports often contain specific numerical information (e.g., electrocardiograms contain measurements such as “QRS duration” and “PR interval”), many different reports contain common measurements (e.g., many reports from both routine and emergency events record patient blood pressure and heart rate), and reports contain diverse other numerical data (e.g., medication dosage). Additionally, EMRs often present numerical data in the context of explicit ranges to aid interpretation. For example, many blood tests measure the concentrations of chemical compounds which are reported with semantic labels, such as “normal” or “elevated,” plus corresponding ranges, such as “$150 - 200$ ppm” or “≥ 200 ppm.”

3.5 Unstructured text

NLP offers many approaches – covering a large range of complexity – for modeling the structure of unstructured text. These include unordered collections of words or more complex strings, collections of entities and relationships between entities and sophisticated structures that explicitly represent a grammar or other generative language model consistent with the text. NLP techniques for identifying different kinds of structure include both domain-agnostic and domain-specific approaches. In our work, we will leverage both language-agnostic (§4.3) and language-specific (§4.4) techniques for viewing unstructured text documents as a bags of words and extensions of this model. We describe these models of structure in text, as well as others that are beyond the scope of our current methodology but plan to study in future work.

The most basic model of text is as a *bag of words* in which a corpus of documents is associated with a vocabulary of N words. Each document is represented as an N-dimensional vector, where each vector element corresponds to one word in the vocabulary; this representation ignores word order. Depending on the application, common choices for the vector elements include (1) ones or zeros that are *indicators* for the presence or absence, respectively, of each word in the document, (2) word *counts* or *frequencies* and (3) more sophisticated quantities such as the *tf-idf weight* (term frequency-inverse document frequency) which balances the frequency of a word within a document against its frequency over the entire corpus [Salton and Buckley 1987]. The bag of words and related models extend naturally to other objects including *n-grams* and meaningful groups of words such as *phrases* or *constituents* (e.g., noun groups and verb groups). These models form the basis of our feature extractors (§4).

Below, we describe additional kinds of structure or features that can be extracted from unstructured text:

—We can derive other representations of text from a bag of words model, such as a *graph* whose nodes are words and (weighted) edges indicate the (strength of) co-occurrence of pairs of words in a corpus of text. a *bipartite graph* whose two node types are documents and words and edges connect each document to the words contained in that document.

—Instead of tracking all of the words, we sometimes only care about certain kinds of words within a text, such as named entities, relationships between entities or words with specific semantic meaning.

—For many applications, we need to understand a text's semantic meaning. For example, we often want to answer questions about text such as “What are the main points?” to generate a summary and “What general semantic topics are involved?” to tag the text with key words.

—We sometimes care about the detailed grammatical structure of text; the choice of representation is governed by the complexity of the rules or grammar that describes the text.

3.6 Structured prose

Not all structure in HR semistructured text is conveyed through visual cues. In the previous section, we considered weak (e.g., n-grams) and implicit (e.g., grammar) forms of structure in unstructured text (§3.5). Through our experience with EMRs, we realized that text can appear visually unstructured while actually being highly structured in the sense of conforming to a restricted language model. We introduce the term *structured prose* to describe such text that is far more constrained than natural language, often involving only a small vocabulary and a simple set of rules.

Structured prose is used to convey information because it is concise and precise: its generative model is limited and produces phrases that are not prone to ambiguity. Interfaces such as drop-down forms and click boxes are common methods for generating structured prose; they enforce the use of a curated vocabulary and limit phrase complexity. A strict interface might be relaxed by allowing human users to optionally add to the vocabulary, e.g., by typing in a text box.

4. METHODS FOR EXTRACTING STRUCTURED FEATURES

In the previous section (§3), we presented different kinds of structure that are commonly found in HR semistructured text. In this section, we describe practical methods for extracting structured features from text and highlight the particular methods that we actually use in our evaluation. We also describe additional practical details and challenges we encountered, some specific to our EMR dataset.

4.1 Our strategy

Our high-level strategy has been to break documents into independent blocks of text and then apply our feature extractors within each block. This is useful for several reasons:

—The blocks have simpler structure than the complete text.

—While all documents in a corpus of HR semistructured text may be distinct, many often share similar or identical blocks of text. This enables the practical optimization of only processing unique text blocks.

—The visual formatting of HR semistructured text results in documents that as a whole do not look like standard natural language text – in contrast to, e.g., newspaper articles. This limits our ability to apply text processing software which typically expects standard natural language input. However, we can break down the text into blocks that in isolation are each closer to natural language.

4.2 Document-level structure

As we mentioned earlier, we view document-level structure as corresponding to the meaningful or logical regions within a document and the relationships between those regions. Our approach thus far has been to extract the former and ignore the latter. In this report, we focus on extracting and exploiting structural features occurring *within* these regions rather than the high-level structure or schema relating them. Future research in this area includes developing methods for both extracting these schema and understanding how applications can leverage them. Some of the existing literature on schema extraction for HTML and XML documents would be relevant to such efforts (§7.2).

We are only interested in the simpler task of breaking up a document into smaller blocks of text for further processing and take a heuristic approach. A naive heuristic is to split the text on line breaks. This can be effective if, for example, each key-value pair appears on separate lines and paragraphs of text contain no internal line breaks. Such an approach was too naive for our EMR dataset, see Appendix A for details about how we horizontally partition EMR text reports. We apply all of our feature extractors (§4.3-§4.4) to these smaller text blocks rather than entire documents.

4.3 Lightweight feature extractors

Here we present lightweight feature extractors that are fast and require minimal domain knowledge. We start with the basic concepts of tokenization and bags of words – the latter is arguably the most naive representation of text – and consider a suite of extractors that build upon this foundation.

4.3.1 *Bags of words.* We can easily derive bags of words from text. The first step is to:

—**Tokenize the input text**. A naive tokenization scheme would be to simply split the text on white spaces. We use a more sophisticated tokenization scheme that uses regular expressions to identify tokens and map each to one of the following *token types*: `date`, `time`, `code`, `number`, `alpha`, `punctuation`. Note that we are using "type" in the programming language sense; in natural language processing, "token types" sometimes refers to distinct tokens but this is not what we mean. We can represent each typed token as 2-tuples, `(token, token_type)`. See Table I, rows B and C for examples of naive tokenization and our typed tokenization. Details of our tokenization scheme are in Appendix C.

A. Original text	`Blood pressure of 100/70`
B. Naive tokens (split on white space)	`Blood, pressure, of, 100/70`
C. Typed tokens	`(Blood, alpha), (pressure, alpha), (of, alpha), (100, number), (/, punctuation), (70, number)`
D. Tokens after mapping to lower case	`blood, pressure, of, 100, /, 70`
E. Tokens after removing stop words	`blood, pressure, 100, /, 70`
F. Tokens after filtering for alphabetic characters	`blood, pressure`
G. Tokens after Porter stemming	`blood, pressur`

Table I. Examples of tokenization and extracting bags of words.

We use these typed tokens later, e.g., to filter tokens by type. As an alternative to the 2-tuple representation of typed tokens, we find it useful to map input text to *typed text* that we represent as a 3-tuple, `(template_text, data_tuple, data_type_tuple)`, where:

(1) `template_text` is a string where we replace all tokens of particular types with special placeholder tokens; in our work we replace tokens of types `date`, `time`, `code` or `number` with the placeholders `$d`, `$t`, `$c` or `$n`, respectively.
(2) `data_tuple` is a tuple of typed values corresponding to the replaced tokens in the `template_text`.
(3) `data_type_tuple` is a tuple representing the data types of the values in `data_tuple`; this field is not strictly necessary but we find it convenient to represent explicitly.

We will exploit this typed text representation in several of our feature extractors, in particular to collapse special tokens (e.g., numbers and dates) (§4.3.2) and to extract quantitative data (§4.3.3). Using our example from Table I, the text ``` ``Blood pressure of 100/70'' ``` maps to this typed text 3-tuple:

```
(``blood pressur $n $n'', (100, 70), (number, number)).
```

Additional, optional steps to normalize the text include the following:

—**Map alphabetic characters to lower case.**
—**Remove stop words.** We always filter out the stop words listed in Appendix B.
—**Remove tokens of particular types.** We always filter out tokens corresponding to the `punctuation` type. We experiment with filtering out all tokens with types of `date`, `time`, `code` or `number`; in other words, we keep only tokens with type `alpha`. We also filter out redacted data denoted by `XXXX` in our dataset.[7]
—**Apply a spell-checker.** We did not include spell checking as a pre-processing step. This would be worthwhile to include in future work.
—**Perform stemming.** We experiment with Porter stemming, an algorithm that removes common morphological and inflexional word endings in English, e.g., `'ed'`, `'ing'`, `'ly'`, etc. [por]

[7]Since EMR data comes from real patients, researchers who analyze EMR data must take great care to maintain patient privacy. One important step is to de-identify the data, which involves removing patient identifiers such as names, alternate names and IDs used within the EMR system. The Regenstreif dataset was de-identified by other parties prior to our analysis of this data according to an aggressive and fairly naive algorithm. The redacted dataset uses the token `'XXXX'` to replace all instances of all substrings matching *any* patient name parts or IDs appearing *anywhere* within the dataset. We excluded any tokens containing the substring `'XXXX'` from our analysis.

Tokenizing an input string of text produces an ordered list of string tokens. A bag of words simply loses or does not make use of the ordering information. After tokenization, the remaining optional steps have the effect of reducing the dimensionality of the dataset, i.e., the total number of distinct extracted tokens.

4.3.2 *Bags of n-grams.* Given a string of input text, our procedure above for producing a bag of words first produced an ordered list of string tokens. We can easily derive n-grams from this list: all we have to do is slide a window of length n over the list and output the tokens in the window. See Table 4.3.2, rows A1-A3 for simple examples. In addition to the optional steps described above for producing bags of words, we experiment with another pre-processing step:

—**Replace tokens of certain types with special tokens.** This corresponds to mapping the input text to the `template_text` described above (§4.3.1). See Table II, rows B1-B4 for simple examples.

A1. Original text	`Blood pressure of 100/70`
A2. 2-grams with lower case alphabetic tokens	`blood pressure, pressure of`
A3. 2-grams after stop word removal and Porter stemming	`blood pressure`
B1. `template_text` after stop word removal and Porter stemming	`blood pressur $n $n`
B2. 2-grams with `template_text` tokens	`blood pressur, pressur $n, $n $n`
B3. 3-grams with `template_text` tokens	`blood pressur $n, pressur $n $n`
B4. 4-grams with `template_text` tokens	`blood pressur $n $n`

Table II. Examples of extracting n-grams.

4.3.3 *Bags of quantitative data.* Since our tokenization scheme associates each token with a particular token type (§4.3.1, Appendix C), we can exploit this information to associate tokens in strings of text – including n-grams, phrases or sentences – with explicitly typed data values.

—**Map text to typed text.** We map input text to typed text represented as 3-tuples, `(template_text, data_tuple, data_type_tuple)`, as described earlier (§4.3.1). We can further explicitly type `number` tokens more precisely, e.g., as `int` or `float`. See Table III, row C for an example.

—**Expand tuples in the typed text representation.** In the above step, we map input text to typed text represented as a single 3-tuple whose first field is a string and whose second and third fields are tuples of some length k corresponding to the number of data values. We can expand this nested representation of the typed text into a series of k tuples, each corresponding to one of the data values. See Table III, row D for an example.

—**Normalize data values.** Each distinct value of `template_text` is associated with a distribution of `data_tuple` values spanning some range, so we may want to normalize these values. For example, we can normalize numerical data values by rescaling them to the interval [0, 1] or mapping them to *z-scores*, the number of standard deviations a particular value is away from the mean of the distribution. As another example, if we recognize multiple date formats, we can standardize them to a single format. See Table III, row E for an example.

—**Discretize data values.** A particular instance of `template_text` may be associated with many distinct `data_tuple` values. By computing basic statistics about the distribution of these values, we can discretize them. For example, using the z-scores (z) described immediately above, we can map the data values to three categories, `high` ($z > 1$), `medium` ($|z| \leq 1$) and `low` ($z < -1$), or to five categories, `very high` ($z > 2$), `high` ($1 < z \leq 2$), `medium` ($|z| \leq 1$), `low` ($-2 \leq z < -1$) and `very low` ($z < -2$). More generally, we can map data values to ranges corresponding to the bins of a histogram computed from the empirical distribution; for data distributed over a logarithmic scale, we might use a histogram of log of the empirical data. See Table III, row F for an example.

4.4 Domain- and language-specific feature extractors

Our strategies that depend on heavier natural language processing techniques make use of MetaMap, a software package developed by the U.S. National Library of Medicine (NLM) at the National Institutes of Health (NIH) [Aronson 2001; 2006]. Before describing our feature extractors, we provide a brief overview of MetaMap's processing pipeline. When MetaMap is given input text, it first performs lexical and syntactic

A. Original text	`Blood pressure of 100/70`
B. Normalized text	`blood pressur 100 70`
C. Typed text	`(blood pressur $n $n, (100, 70), (int, int))`
D. Expanded typed text tuples	`(blood pressur $n $n, 1, 100, int)` `(blood pressur $n $n, 2, 70, int)`
E. Expanded and normalized to z-scores	`(blood pressur $n $n, 1, -1.05)` `(blood pressur $n $n, 2, -0.43)`
F. Expanded and discretized to high/medium/low	`(blood pressur $n $n, 1, low)` `(blood pressur $n $n, 2, medium)`

Table III. Examples of quantitative feature extraction.

analysis including: (1) tokenization, sentence splitting and acronym and abbreviation detection, (2) part-of-speech tagging, (3) lexical lookup and (4) syntactic analysis. The second half of MetaMap's pipeline performs: (5) variant generation, (6) candidate identification, (7) mapping construction via the Unified Medical Language System (UMLS) and (8) word-sense disambiguation. The UMLS includes a dictionary of concepts, a thesaurus mapping these concepts onto preferred names and a semantic tree that places these concepts within broader semantic categories. MetaMap's different processing steps produce rich output that can be used to extract many kinds of features. We highlight some of these through examples.

Table IV shows features that come from MetaMap's lexical and syntactic analysis steps. For the input phrase ``` ``sinus rhythm has replaced electronic atrial pacemaker'' ``` (row A), we show how MetaMap breaks this into phrases (row B) and further into constituents (row C). Each constituent maps to a lexical category (e.g., noun or adjective, row D) and a syntax type (e.g., head or modifier, row E.).

Feature	**Example**
A. Input text	sinus rhythm has replaced electronic atrial pacemaker
B. Phrases	sinus rhythm, has, replaced electronic atrial pacemaker
C. Constituents	sinus rhythm, has, replaced, electronic, atrial, pacemaker
D. Lexical categories (matched to constituents)	(sinus rhythm, noun), (has, aux), (replaced, adj), (electronic, adj), (atrial, adj), (pacemaker, noun)
E. Syntax types (matched to constituents)	(sinus rhythm, head), (has, aux), (replaced, mod), (electronic, mod), (atrial, mod), (pacemaker, head)

Table IV. Examples of features extracted via MetaMap's lexical and syntactic analysis pipeline.

Table V gives a few more examples of features we can extract via MetaMap's lexical and syntactic analysis. By combining phrase and constituent features, we can for example extract noun phrases, i.e., phrases containing a head noun plus optional modifiers. We italicize constituents corresponding to head nouns within phrases. Breaking down the inputs into feature vectors of phrases or constituents suggest different ways to compare or align related inputs (rows A-B and C-D).

	Input text	**Phrases identified by MetaMap**
A.	electronic atrial pacemaker	electronic atrial *pacemaker*
B.	electronic ventricular pacemaker	electronic ventricular *pacemaker*
C.	ST \T\ T wave abnormality, consider anterolateral ischemia or digitalis effect	*st t t wave abnormality*, consider, *anterolateral ischemia*, or, *digitalis effect*

Table V. Phrases identified by MetaMap, where we have italicized constituents that are head nouns

Table VI shows features corresponding to MetaMap's final output for the input phrase ``` ``sinus rhythm'' ```. The Meta Candidates (rows A) show possible mappings between substrings, e.g., constituents or tokens, and concepts in MetaMap's dictionary. Each concept maps to a preferred name via MetaMap's thesaurus and is a member of a broader semantic category. Each Meta Candidate is given a score, with 1000 being the maximum possible score. The Meta Mappings (rows B) represent the final mappings of the entire phrase. Each Meta Mapping is given a score, again with 1000 being the maximum.

Table VII illustrates MetaMap's ability to map different inputs onto the same preferred name (rows A-C). While some abbreviations are accepted (row C), others are not (rows D-E); notice that the latter receive lower scores because their Meta Mappings only correspond to part of the input.

Output name	Concept	Preferred name	Semantic category	Score
A. Meta Candidates	Sinus rhythm	Sinus rhythm	Finding	1000
	sinus rhythm	electrocardiogram rhythm strip 3-lead: sinus rhythm	Finding	1000
	Sinus	pathologic fistula	Anatomical Abnormality	861
	SINUS	Nasal sinus	Body Space or Junction	861
	Rhythm	Rhythm	Finding	861
	Sinus	Sinus - general anatomical term	Body Space or Junction	861
	rhythm	physiologic rhythm	Physiologic Function	861
	Rhythmicities	Periodicity	Temporal Concept	761
B. Meta Mappings	Sinus rhythm	Sinus rhythm	Finding	1000
	sinus rhythm	electrocardiogram rhythm strip 3-lead: sinus rhythm	Finding	1000

Table VI. Examples of features extracted via MetaMap's full pipeline for the input phrase ‘‘`sinus rhythm`’’.

	Input text	Meta Mapping	Preferred name	Semantic category	Score
A.	heart attack	Heart attack	Myocardial Infarction	Disease or Syndrome	1000
B.	myocardial infarction	Myocardial Infarction	Myocardial Infarction	Disease or Syndrome	1000
C.	myocardial infarct	Myocardial Infarction	Myocardial Infarction	Disease or Syndrome	1000
D.	myocardial infarc	Myocardial	Myocardium	Tissue	694
E.	myocard infarct	Infarct	Infarction	Pathologic Function	861

Table VII. Examples of MetaMap's output for similar inputs.

Thus with MetaMap, we can directly produce:

—**Bags of phrases or constituents.** Notice that in contrast to bi-grams and higher order n-grams, these are non-overlapping grammatical objects. As before, we can apply optional normalization steps such as stemming and stop word removal (§4.3.1) to further simplify these bags. Depending on our feature extraction task, we could also use the lexical or syntax information to filter the data, e.g., filter for nouns and adjectives.

—**Bags of Meta Candidates or Meta Mappings.** We note that each of the Meta Candidates and Mappings is associated with a unique code which makes this output easier to work with. We can use the associated scores and semantic categories to filter this data, e.g., filter for the highest scoring Meta Mappings belonging to the "Finding" semantic type.

We note that while MetaMap does not recognize many tokens including numbers, we can combine features extracted via MetaMap with our methods for extracting typed data to produce "typed Meta Mappings". For example, when MetaMap processes the input text ‘‘`blood pressure 100/70 mmHg`’’, it recognizes the entire input as a single phrase but its Meta Mappings only include the concepts for ‘‘`blood pressure`’’ and ‘‘`mmHg`’’, but we can extract the numerical data and bind it to this output to produce a composite feature, e.g., `((blood pressure, mmHg), (100, 70), (int, int))`.

5. EVALUATION

We show that explicitly capturing this structure enables both increased accuracy in statistical analytic tasks (§5) and a broad range of applications and user interfaces (§6). In this section, we compare the performance of various sets extracted features against the most naive text features, i.e., bags of words, in a specific data analytic case study.

5.1 Case study objective

A group of epidemiologists is studying the protective effect of high-intensity versus normal-intensity statins, a class of cholesterol-lowering drugs, against composite cardiovascular events: myocardial infarction (heart attack), acute coronary syndrome (obstruction of coronary arteries) and stroke. This study is analogous to one of the use cases for structured EMR data described earlier (§2.3). In particular, this is a retrospective study where one of the major challenges is identifying and adjusting for confounding factors that are associated with both *exposure* (i.e., whether the patient received a high-intensity or normal-intensity statin) and *outcome* (i.e., whether or not the patient experienced a composite cardiovascular after beginning statin treatment). Clinical trial results show that high-intensity statins are slightly more protective than low-intensity

statins; our high-level objective is to show whether we can replicate these results in our retrospective study by identifying and adjusting for the right confounders.

Most epidemiology studies like this one utilize highly-structured information to identify confounders – such as coded medical claims data (e.g., prescribed drugs and treatments denoted by unique identifiers) and structured EMR data (e.g., patient demographic information) – but ignore the rich semi-structured and unstructured text information also contained in EMR text reports. In our evaluation, we study the added value of extracting features from EMR text reports and the relative value of different sets of extracted features in the high-dimension propensity score (hd-PS) algorithm, a particular method for identifying and adjusting for confounders [Schneeweiss et al. 2009].

5.2 Regenstrief dataset

Our cohort includes 9,906 individuals in the Regenstrief Medical Record System / Indiana Network for Patient Care (RMRS/INPC), a regional EMR system in central Indiana, from 1 July 2004 to 31 May 2010. For each of these individuals, we include medical claims and EMR data in our study as follows:

—The study begins with a 12-month period of time during which the individual has no fills for statins, followed by a variable period of time during which the patient uses a particular *index statin*.

—The study continues until either the patient either (1) discontinues use of the index statin, (2) a composite cardiovascular event occurs in which the first such event occurs at least 30 days after statin initiation or (3) the study period ends.

—A patient can re-enter the study by satisfying the start criteria again.

The resulting dataset contains 607,668 EMR text reports.

5.3 Results

We summarize the effect of various feature sets on the hd-PS algorithm results by reporting an *odds ratio* (OR) comparing the odds of an outcome x (composite cardiovascular event) given one treatment (high-intensity statins) versus another (normal-intensity statins), where the *odds* of outcome x are given by $P(x)/(1-P(x))$. The previous clinical trials data showing that high-intensity statins have a slightly more protective effect means that we should obtain an OR slightly lower than 1.0.

We compare our results against: (1) no adjustment for any confounders which produces an OR of 2.19 with a 95% confidence interval of 1.72 to 2.80, (2) basic adjustment for the known confounders age and sex which gives an OR of 2.05 (1.58, 2.64) and (3) adjustment for age and sex plus confounders identified from *structured* medical claims data which reduced the OR to 1.21 (0.86, 1.71). The crude OR is much higher than the expected OR based on clinical trials, reflecting the presence of confounders. The age/sex adjusted OR is somewhat lower than the crude OR, reflecting that age and sex are indeed confounders, but they don't account for all confounders in the data. The structured medical claims data contains many confounders that drive down the OR significantly. Below, we evaluate whether the *unstructured* EMR text reports contain additional confounders that further reduce the OR. Our results are summarized in Table VIII.

Our most naive set of features extracted from the EMR text reports corresponds to a simple bag of words representation. As described earlier (§4.3.1), we tokenize by according to the scheme described in Appendix C and remove the stop words listed in Appendix B. Adjusting for age/sex, confounders from the structured claims data and words (unigrams) from the unstructured EMR text reduces the OR to 1.09 (0.76, 1.57). Performing Porter stemming on the words as well as replacing numerical tokens – numbers, dates, times and codes – with special tokens appears to result in a slightly but not significantly higher OR; we observe this in our results for all n-gram lengths.

Next, we adjusted for age/sex, confounders from the structured claims data and n-grams from the unstructured EMR text for $n = 2, 3, ..., 7$. Bigrams without stemming perform similarly compared to unigrams, producing an OR of 1.03 (0.71, 1.49). Higher order n-grams also perform similarly, producing OR results close to but above 1.0.

Our results show that features extracted from unstructured EMR text reports reveal additional confounders not present in the structured claims data. In order to obtain more significant and definitive results, we would require larger a dataset involving more patients. We note that the medical claims data and EMR text reports contain overlapping information; our results would have appeared more dramatic if we had not included confounders from the medical claims data in our comparison.

Model	Porter stemming	Numbers replaced	OR (95% CI)
Crude	-	-	2.19 (1.72, 2.80)
Age/sex adjusted	-	-	2.05 (1.58, 2.64)
Age/sex + claims adjusted	-	-	1.21 (0.86, 1.71)
Age/sex + claims + 1-gram adjusted	✗	✗	1.09 (0.76, 1.57)
	✓	✗	1.16 (0.81, 1.66)
	✓	✓	1.14 (0.80, 1.63)
Age/sex + claims + 2-gram adjusted	✗	✗	1.03 (0.71, 1.49)
	✓	✗	1.22 (0.85, 1.76)
	✓	✓	1.24 (0.86, 1.77)
Age/sex + claims + 3-gram adjusted	✗	✗	1.08 (0.75, 1.57)
	✓	✗	1.09 (0.76, 1.57)
	✓	✓	1.15 (0.80, 1.64)

Table VIII. Odds ratio (OR) for various models reported with 95% confidence intervals (CI). The symbols ✗and ✓denote "without" and "with," respectively. "Numbers replaced" denotes whether tokens corresponding to numbers, dates, times and codes were replaced by special tokens. Results with 4-grams through 7-grams are similar to those with 2-grams and 3-grams.

6. APPLICATIONS

In this section, we present example applications that can be quickly built on top of a substrate of features extracted from EMR text reports. We focus on applications enabled by automatically extracting quantitative features from a smaller dataset of 7,083 EMR text reports from the Partners HealthCare Longitudinal Medical Record (LMR) system in Massachusetts. Like the Regenstrief data, this dataset comes from a cohort of cardiac patients but includes any EMR entries, not just cardiology-related reports.

We horizontally partitioned each ECG report (Appendix A) and focused on those resulting blocks of text containing numbers as identified via our tokenization scheme (Appendix C). We transformed each of these text blocks into our typed text 3-tuple representation presented earlier (§4.3.1) and expanded tuples containing more than one data value (§4.3.3). For each distinct instance of `template_text`, we automatically computed basic statistics characterizing the distribution of observed data values including its mean, variance and summary as characterized by a 20-bin histogram.

For illustration purposes, we focus on the most prevalent kind of EMR entries in our dataset, the set of all 2001 electrocardiogram (ECG) text reports. The first and second examples of EMR entries that we showed (§1.3) are representative of these ECG reports.

By extracting typed text and identifying the distinct instances of `template_text`, we were able to quickly detect inconsistencies and errors in the reports. For example, we detected these very similar instances: `P-R-T AXES $n $n $n` and `P-R-T AXES $n $n`. It turns out that while the latter occurred a substantial but small number of the reports, it represents ambiguous and incomplete information likely due to human data entry error. We imagine an EMR system that could instantly prevent data entry errors by flagging anomalous, i.e., low-frequency, instances of `template_text` and suggesting close alternatives in the spirit of a spell-checker.

From the ECG reports, our extraction scheme identified data tuples corresponding to standard measurements used to characterize an ECG signal. Figure 1 shows examples of one-dimensional histograms automatically plotted from this extracted data. We highlight outliers that might correspond to errors due to human error in data entry, bugs in data analysis software or environmental interference causing faulty machine behavior. This kind of outlier detection could be incorporated into EMR systems to automatically detect anomalous data in real time, thus enabling one of our use cases for structured EMR data (§2.2).

For each pair of distinct `template_text` instances, we also automatically generated a scatterplot of all data values. Figure 2 shows examples of these two-dimensional scatter plots. We again highlight outliers, including a very suspicious apparent functional dependency between `VENT. RATE` and `PR INTERVAL`. We would not expect such a functional dependency to arise from random human data entry error and thus suspect a problem with the ECG machine hardware or signal processing software.

7. RELATED WORK

Our work uses and builds upon methods and applications of representing or extracting features from unstructured (§7.1) and semistructured text (§7.2) as well as medical text in particular (§7.3).

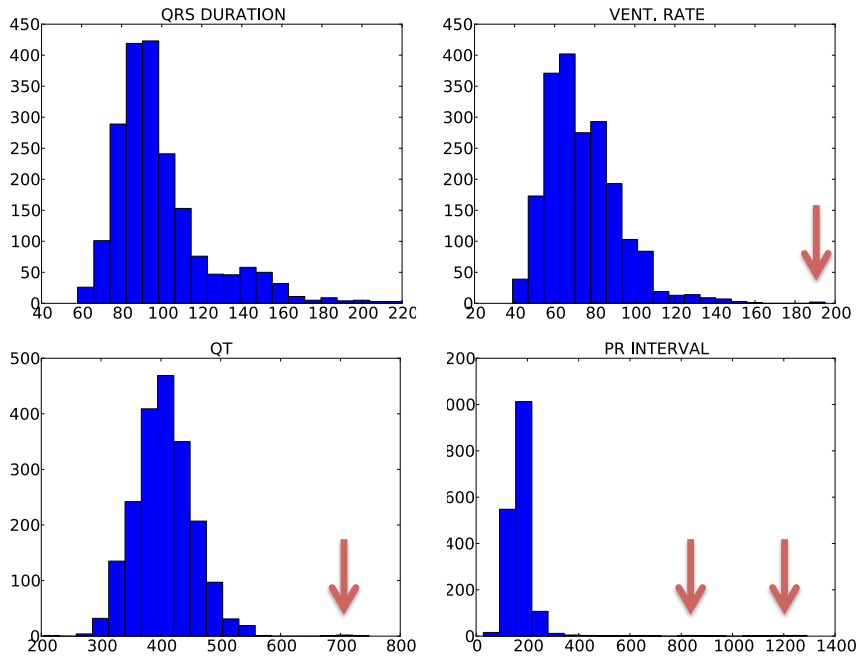

Fig. 3. One-dimensional histograms of quantitative measurements automatically extracted from a corpus of electrocardiogram (ECG) text reports. QRS duration (top left), ventricular rate (top right), QT interval (bottom left), PR interval (bottom right). The red arrows point to groups of outliers.

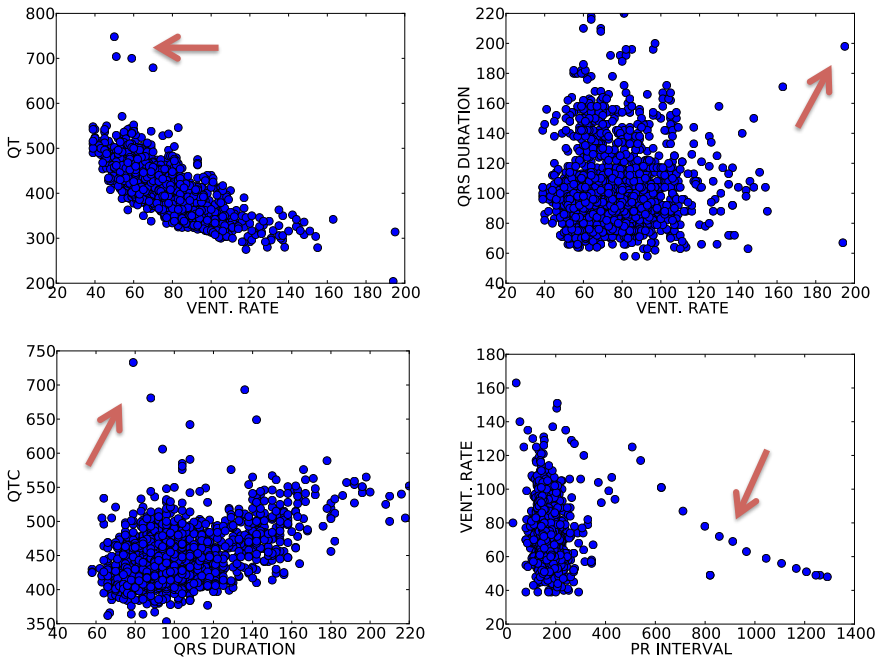

Fig. 4. Two-dimensional scatter plots of quantitative measurements automatically extracted from a corpus of electrocardiogram (ECG) text reports. Clockwise from top left: QT interval vs. ventricular rate (top left), QRS duration vs. ventricular rate (top right), QTc interval vs. QRS duration (bottom left), ventricular rate vs. PR interval (bottom right). The red arrows point to groups of outliers.

7.1 Unstructured text

Natural language processing (NLP) and information retrieval (IR) techniques for working with unstructured text are surveyed extensively elsewhere [Jurafsky and Martin 2008; Manning et al. 2008]. NLP in particular offers many more sophisticated models of language that capture grammatical structure in unstructured text. Here, we highlight some basic methods and applications for extracting structured features from text.

Modeling unstructured text documents as bags of words or as n-grams serves as a foundation for many tasks such as categorizing documents with respect to one or more topics [Damashek 1995; Blei et al. 2003; Wang et al. 2007], modeling word order in natural language [Brown et al. 1992] and and determining authorship [Mosteller and Wallace 1964; Keselj et al. 2003]. Our work relied heavily on techniques related to bags of words and n-grams. We note that while the technique of replacing tokens such as numbers and dates with special tokens is commonly used to decrease the number of distinct tokens, we do not know of prior work similar to ours for investigating distributions of data values associated with tokens or n-grams (§4.3.3).

McCallum provides an overview of information extraction techniques for obtaining structured data from unstructured text, breaking them down into a series of four tasks: (1) segmentation to break down text into smaller pieces, e.g., meaningful sections or phrases, (2) classification to assign attribute labels or database fields to the segmented text pieces, e.g., "title", "price", (3) association to determine which fields should be represented together in the same field, e.g., separate paragraphs that were broken apart by segmentation but belong together and (4) normalization and deduplication to standardize formats and remove redundant records [McCallum 2005]. Our horizontal partitioning of EMR reports (Appendix A) is an example of segmentation, as is breaking text into phrases or constituents (§4.4). Our view of semistructured text as containing both metadata and data values informs how we approach the IR classification task. We have not been concerned with applications that depend on association. Many of our feature extractors perform some kind of normalization: dimension reduction techniques such as stemming and stop word removal, our methods for normalizing and discretizing quantitative data values and our use of MetaMap's Metathesaurus to map synonyms to the same coded concept are all examples of normalization.

Typical IR tasks include (named) entity recognition, e.g., to populate database records from unstructured text [Wick et al. 2006], event extraction, e.g., biomedical events from academic literature [Riedel and McCallum 2011] and relationship discovery, e.g., to transform unstructured text into resource description framework (RDF) relations [Ramakrishnan et al. 2006]. We plan to build future feature extractors based on these IR tasks.

Note that the MetaMap software stack relies heavily on many NLP techniques, some of which we already summarized briefly (§4.4) and others we mention below (§7.3).

7.2 Semistructured text

As we noted earlier (§1), Buneman introduced the notion of semistructured data [Buneman 1997]. The vast majority of research in extracting structured information from semistructured data or semistructured text has been focused on Web pages. Some representative paper titles that sound quite general but are in fact about extracting data from HTML or XML include "Efficient Substructure Discovery from Large Semi-structured Data" [Asai et al. 2002] and "Scalable Attribute-Value Extraction from Semi-structured Text" [Wong et al. 2009].

In a survey on Web mining, Kosala and Blockee organize data extraction research into information retrieval (IR) and database approaches [Kosala and Blockeel 2000]. In the first view, we want to perform traditional IR tasks with Web data. These IR tasks rely on representing Web content using structures such as bags of words, n-grams, terms, phrases, concepts, ontologies and relations. In latter perspective, we view Web data as belonging in a database that we would like to query and rely on structuring data into relational tables or edge-labeled graphs. Applications of these two views are concerned with identifying different kinds of structure in Web data: information retrieval applications seek to uncover semantically meaningful entities and relationships while database applications seek to populate records in a database and rely on schema discovery. The information retrieval and database views do not represent completely disjoint approaches, but provide a useful framework for understanding Web data extraction research from different communities.

A program for extracting structured data from Web pages is called a *wrapper*, and wrapper generation methods fall into three categories: manual, semi-automatic, and automatic. Early research developed languages to facilitate manual wrapper generation, and are surveyed in an overview of Web data extraction tools [Laender et al. 2002]. In our work, we avoid manual techniques for extracting structured data but note

that it is standard practice for medical informatics companies to manually write custom wrappers or parsers in order to structure data from hospitals or medical insurers [cite interview]. More recent work in Web data extraction has focused on automatic wrapper generation and has already been surveyed extensively [Liu 2011]. Most existing approaches exploit tag tree structure and include methods such as tree matching and partial tree alignment [Zhai and Liu 2006]. If we can discover tree structures, e.g., by learning simple grammars, that describe the schema of related HR semistructured text documents, then we expect to be able to apply or modify tree-based techniques for automatically extracting structured data from Web pages.

Tables are a specific kind of structure found in semistructured text. Research in extracting tables from – using our terminology – HR semistructured text includes methods based on heuristics [Pyreddy and Croft 1997] and conditional random fields [Pinto et al. 2003]. We plan to use these methods in future work to extract tables from EMR reports and other examples of HR semistructured text such as government reports. Other areas of table extraction research include identifying tables plus their metadata in PDF documents [Liu et al. 2007] and extracting tables from HTML and XML by exploiting `<table>` tags [Tengli et al. 2004], visual redering in the browser [Gatterbauer et al. 2007] and at the scale of billions of tables via the WebTables system [Cafarella et al. 2008].

7.3 Medical text

Related work in extracting information from medical text is mostly based on NLP and IR methods. First, we first note that extracting information from academic biomedical text is a related but different problem that exploits the curated and structured nature of databases for biomedical literature; surveys of this research include reviews of text mining in biomedical [Rodriguez-Esteban 2009], genomics [Shatkay and Feldman 2003] and pharmacogenomics [Garten et al. 2010]. Medical and biomedical text mining research appears focused on text-oriented techniques and applications; there does not seem to be much existing research on automatically extracting quantitative data from this text. We highlight one exception concerned with mining biomedical literature for pharmacokinetic data [Wang et al. 2009].

Below, we focus on research in mining medical text such as EMRs; some existing survey topics include text mining in healthcare [Raja et al. 2008], health informatics [Bath 2008] and extracting information from EMR free text [Meystre et al. 2008]. As discussed earlier, our feature extractors specific to medical text depend on the MetaMap software package for mapping biomedical text to the Unified Medical Language System (UMLS) Metathesaurus [Aronson 2001; 2006].

Especially in medical text, sentiment and context are important for understanding semantic meaning. MetaMap includes an implementation of the NegEx algorithm to perform negative sentiment analysis in clinical text. NegEx uses regular expressions to identify negation triggers, e.g., "not" and their target phrases [Chapman et al. 2001]. In our future work with EMR data, we plan to evaluate and potentially incorporate negative sentiment analysis into our feature extractors. ConText is an extension of the NegEx algorithm for identifying contextual features from clinical text, e.g., "whether a condition is negated, historical or hypothetical, or experienced by someone other than the patient" [Chapman et al. 2007].

In contrast to our work, previous applications of medical data mining tend to either rely on structured or coded information, e.g., to characterize morbidity using ICD-9 codes [Schildcrout et al. 2010], or only tackle small, focust datasets, e.g., 125 patient records concerning breast complaints [Zhou et al. 2006]. Other example research areas in the medical text domain include extracting medical problems from EMRs [Meystre and Haug 2006], identifying the main topic and keywords in clinical questions via supervised machine learning techniques [Cao et al. 2010] and medical data mining on the Web [MacLean and Seltzer 2011].

8. DISCUSSION

We have described the challenge of extracting structured features from HR semistructured text and presented a framework for confronting this challenge. Our approach thus far has been guided by an in-depth case study involving a large amount of real data from the medical domain. Our first steps have included evaluating a suite of text-oriented approaches from the information retrieval and natural language processing communities. We plan to pursue future work that expands our current research in several major directions:

—**Additional feature extractors.** We will study methods to recover more highly-structured, document-level information from HR semistructured text to infer more sophisticated schema and hierarchical structure for organizing and relating extracted features. While we expect these methods will enrich direct human interaction with extracted data, it is currently an open question whether these features will aid in

statistics and machine learning tasks. To evaluate these methods, we will continue our case study based around working with epidemiologists interested in exploiting features extracted from EMR text reports to perform retrospective drug effectiveness studies. We will follow our evaluation strategy (§5) to compare results obtained with any new methods against methods presented here to measure whether additional feature sets further improve results with hd-PS.

—**EMR-specific applications.** Using the features that we have already extracted, we plan to develop a suite of different applications aimed at helping (1) EMR systems automatically detect anomalous trends and outliers potentially corresponding to outbreaks, faulty equipment or human error, (2) medical professionals interact with a patient's EMR data, e.g., by automatically generating a summary view that paints a picture of the patient's medical history, (3) medical researchers explore and mine EMR data through intuitive visualization and interaction tools. We also plan to investigate additional data analytic tasks that could benefit from structured features extracted from EMR text reports, such as machine learning algorithms designed to discover meaningful patient subpopulations for classification tasks.

—**Other domains.** We plan to study feature extraction for HR semistructured text in other domains such as government reports, patent law and digital humanities.

9. ACKNOWLEDGEMENTS

We thank (1) our collaborators Jeremy Rassen, Sebastian Schneeweiss and Peter Wahl from the Division of Pharmacoepidemiology at Brigham and Women's Hospital (BWH) and Marc Rosenman from Regenstrief Institute, Inc., (2) committee members Stephen Chong, Jeremy Rassen, Margo Seltzer and Stuart Shieber, (3) Robert Giugliano and Andrea Crompton from the TIMI Study Group at BWH for showing us how cardiologists interact with the Partners HealthCare LMR system, (4) Varun Kanade, Heather Pon-Barry, Elif Yamangil and members of the Systems Research at Harvard (SYRAH) group for valuable feedback.

APPENDIX

A. HORIZONTALLY PARTITIONING EMR TEXT REPORTS

In working with data from multiple EMR systems, we found that splitting on line breaks was too naive because many – although not all – regions of verbose unstructured text have line breaks inserted for visual formatting purposes. Thus in practice, we first used some simple heuristics to identify and remove line breaks that appear to be in the middle of sentences or paragraphs, and then we were able to split the text of each EMR text report. Additionally, we found that the MetaMap software used in §4.4 could not parse many raw EMR text entries because they do not conform to its expectations for input text, i.e., a series of sentences that can be split apart and individually parsed according to its model for natural language.

B. STOP WORD LIST

We removed stop words for our n-gram and Tango chunking analysis. We wanted a fairly conservative list of stop words, so we used the highest frequency words from Project Gutenberg as a guide [wik]. The words with frequency higher than 0.005, ordered by descending frequency are: the, of, and, to, in, i, that, was, his, he, it, with, is, for, as, had, you, not, be, her, on, at, by. We started with this as a base list of stop words; our actual list includes several modifications:

(1) exclude "i" because single characters appear in many medical abbreviations,
(2) exclude "not" because of its semantic importance in expressing negative sentiment and
(3) include "she" for symmetry with respect to "he."

C. TOKENIZATION

Our tokenization scheme applies a series of regular expressions to input text to first identify a special set of tokens – common formats for dates, times, numerical values and identification codes – and then distinguish alphabetic strings from remaining (non-whitespace) characters, i.e., punctuation. We use this process to associate each token with one of the following token types: `date`, `time`, `code`, `number`, `alpha`, `punctuation`. Below is a list of our regular expressions written for use with Python's `re` module [re 2012]:

—`\d1,2/\d1,2/\d4|\d1,2/\d1,2/\d2` for common date formats,

—\d+:\d+([:]\d+)? for common time formats,

—[0-9]1,[a-zA-Z]1,[0-9]1,[0-9a-zA-Z]*|[a-zA-Z]1,[0-9]1,[a-zA-Z]1,[0-9a-zA-Z]* for common codes containing a mix of alphabetic and numeric characters,

—[-]?\d+[\.]?\d* to identify numbers, including negative and decimal numbers,

—\w+ to identify alphabetic tokens.

REFERENCES

Porter stemming algorithm. http://nltk.googlecode.com/svn/trunk/doc/api/nltk.stem.porter-module.html.

Wiktionary:frequency lists. http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2006/04/1-10000.

2009. Health information technology for economic and clinical health (HITECH) act. *Title XIII of Division A and Title IV of Division B of the American Recovery and Reinvestment Act of 2009 (ARRA) Pub. L.,* 111-5 (Feb.), codified at 42 U.S.C. §§300jj et seq.; §§17901 et seq.

2010. Google Books Ngram Viewer. http://books.google.com/ngrams.

2012. re - regular expression operations. http://docs.python.org/library/re.html.

Aronson, A. R. 2001. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. *Proc AMIA Symp*, 17–21.

Aronson, A. R. 2006. MetaMap: Mapping text to the UMLS Metathesaurus. *Text*, 1–26.

Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., and Arikawa, S. 2002. Efficient substructure discovery from large semi-structured data. In *SDM*, R. L. Grossman, J. Han, V. Kumar, H. Mannila, R. Motwani, R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, Eds. SIAM.

Bath, P. A. 2008. Health informatics: Current issues and challenges. *Journal of Information Science 34,* 4, 501–518.

Blei, D. M., Ng, A. Y., and Jordan, M. I. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res. 3*, 993–1022.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. 1992. Class-based n-gram models of natural language. *Comput. Linguist. 18,* 4 (Dec.), 467–479.

Buneman, P. 1997. Semistructured data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. PODS '97. ACM, New York, NY, USA, 117–121.

Cafarella, M. J., Halevy, A. Y., Wang, D. Z., Wu, E., and Zhang, Y. 2008. WebTables: exploring the power of tables on the web. *PVLDB 1,* 1, 538–549.

Cao, Y., Cimino, J. J., Ely, J. W., and Yu, H. 2010. Automatically extracting information needs from complex clinical questions. *Journal of Biomedical Informatics 43,* 6, 962–971.

Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., and Buchanan, B. G. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of biomedical informatics 34,* 5 (Oct.), 301–310.

Chapman, W. W., Chu, D., and Dowling, J. N. 2007. Context: an algorithm for identifying contextual features from clinical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*. BioNLP '07. Association for Computational Linguistics, Stroudsburg, PA, USA, 81–88.

Damashek, M. 1995. Gauging similarity with n-grams: Language-independent categorization of text. *Science 267,* 5199, 843–848.

Garten, Y., Coulet, A., and Altman, R. 2010. Recent progress in automatically extracting information from the pharmacogenomic literature. *Pharmacogenomics 11,* 10, 1467–89.

Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., and Pollak, B. 2007. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th international conference on World Wide Web*. WWW '07. ACM, New York, NY, USA, 71–80.

Jurafsky, D. and Martin, J. H. 2008. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall.

Keselj, V., Peng, F., Cercone, N., and Thomas, C. 2003. N-gram-based author profiles for authorship attribution. *Computational Linguistics 3*, 255264.

Kosala, R. and Blockeel, H. 2000. Web mining research: A survey. *SIGKDD Explorations 2,* 1, 1–15.

Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S., and Teixeira, J. S. 2002. A brief survey of web data extraction tools. *SIGMOD Rec. 31*, 84–93.

Liu, B. 2011. Structured data extraction: Wrapper generation. In *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications. Springer Berlin Heidelberg, 363–423.

Liu, Y., Bai, K., Mitra, P., and Giles, C. L. 2007. Tableseer : Automatic table metadata extraction and searching in digital libraries categories and subject descriptors. *Information Sciences*, 91–100.

MacLean, D. and Seltzer, M. I. 2011. Mining the web for medical hypotheses - a proof-of-concept system. In *HEALTHINF*, V. Traver, A. L. N. Fred, J. Filipe, and H. Gamboa, Eds. SciTePress, 303–308.

Manning, C. D., Raghavan, P., and Schtze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

McCallum, A. 2005. Information extraction: Distilling structured data from unstructured text. *Queue 3,* 9 (Nov.), 48–57.

Meystre, S. and Haug, P. J. 2006. Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation. *J. of Biomedical Informatics 39,* 6 (Dec.), 589–599.

Meystre, S. M., Savova, G. K., Kipper-Schuler, K. C., and Hurdle, J. F. 2008. Extracting information from textual documents in the electronic health record: a review of recent research. *Yearbook of medical informatics 47,* Suppl 1, 128–144.

Mosteller, F. and Wallace, D. 1964. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley.

Pinto, D., McCallum, A., Wei, X., and Croft, W. B. 2003. Table extraction using conditional random fields. In *DG.O*.

Pyreddy, P. and Croft, W. B. 1997. Tintin: a system for retrieval in text tables. In *Proceedings of the second ACM international conference on Digital libraries*. DL '97. ACM, New York, NY, USA, 193–200.

Raja, U., Mitchell, T., Day, T., and Hardin, J. M. 2008. Text mining in healthcare. Applications and opportunities. *Journal of healthcare information management : JHIM 22,* 3, 52–56.

Ramakrishnan, C., Kochut, K., and Sheth, A. P. 2006. A framework for schema-driven relationship discovery from unstructured text. In *International Semantic Web Conference*, I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, Eds. Lecture Notes in Computer Science, vol. 4273. Springer, 583–596.

Riedel, S. and McCallum, A. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Association for Computational Linguistics, Stroudsburg, PA, USA, 1–12.

Rodriguez-Esteban, R. 2009. Biomedical text mining and its applications. *PLoS Comput Biol 5,* 12 (12), e1000597.

Salton, G. and Buckley, C. 1987. Term weighting approaches in automatic text retrieval. Tech. rep., Ithaca, NY, USA.

Schildcrout, J. S., Basford, M. A., Pulley, J. M., Masys, D. R., Roden, D. M., Wang, D., Chute, C. G., Kullo, I. J., Carrell, D., Peissig, P., Kho, A., and Denny, J. C. 2010. An analytical approach to characterize morbidity profile dissimilarity between distinct cohorts using electronic medical records. *J. of Biomedical Informatics 43,* 6 (Dec.), 914–923.

Schneeweiss, S., Rassen, J. A., Glynn, R. J., Avorn, J., Mogun, H., and Brookhart, M. A. 2009. High-dimensional propensity score adjustment in studies of treatment effects using health care claims data. *Epidemiology (Cambridge, Mass.) 20,* 4 (July), 512–522.

Shatkay, H. and Feldman, R. 2003. Mining the biomedical literature in the genomic era: An overview. *Journal of Computational Biology 10,* 6, 821–855.

Tengli, A., Yang, Y., and Ma, N. L. 2004. Learning table extraction from examples. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA.

Wang, X., McCallum, A., and Wei, X. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*. ICDM '07. IEEE Computer Society, Washington, DC, USA, 697–702.

Wang, Z., Kim, S., Quinney, S. K., Guo, Y., Hall, S. D., Rocha, L. M., and Li, L. 2009. Literature mining on pharmacokinetics numerical data: A feasibility study. *J. of Biomedical Informatics 42,* 4 (Aug.), 726–735.

Wick, M., Culotta, A., and Mccallum, A. 2006. Learning field compatibilities to extract database records from unstructured text. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, 603–611.

Wong, Y. W., Widdows, D., Lokovic, T., and Nigam, K. 2009. Scalable attribute-value extraction from semi-structured text. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*. ICDMW '09. IEEE Computer Society, Washington, DC, USA, 302–307.

Zhai, Y. and Liu, B. 2006. Automatic wrapper generation using tree matching and partial tree alignment. In *AAAI*. AAAI Press.

Zhou, X., Han, H., Chankai, I., Prestrud, A., and Brooks, A. 2006. Approaches to text mining for clinical medical records. In *Proceedings of the 2006 ACM symposium on Applied computing*. SAC '06. ACM, New York, NY, USA, 235–239.