ML-Driven Clinical Documentation

by

Divya Gopinath

B.S. Computer Science and Engineering
Massachusetts Institute of Technology, 2019

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author .....
Department of Electrical Engineering and Computer Science
June 30, 2020

Certified by .....
David Sontag
Associate Professor
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

ML-Driven Clinical Documentation

by

Divya Gopinath

Submitted to the Department of Electrical Engineering and Computer Science
on June 30, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Electronic health records (EHRs) have irrevocably changed the practice of medicine by systematizing the collection of patient-level data. However, clinicians currently spend more time documenting information in EHRs than interacting directly with patients, and have adapted to time-intensive note-writing by authoring free-text notes overloaded with jargon and acronyms. Clinical notes are therefore difficult to parse and largely unstructured. This negatively impacts the ability of EHR systems to convey information between different clinicians and institutions, to communicate medical findings to patients, and to allow for programmatic ingestion of data to derive further automatically-learned insights. In this thesis, we present a new EHR system that addresses these problems by using novel machine learning methods to streamline the processes by which clinicians enter in new information and surface relevant details from past medical records. Our intelligent interface aids physicians as they type, allowing for automatic suggestion and live-tagging of clinical concepts to alleviate documentation burden, while simultaneously enabling clinical decision support and contextual information synthesis. Furthermore, as clinicians craft notes we automatically structure and curate their free-text inputs, allowing for further data-driven innovation and improvement. This EHR can reduce physician burnout, decrease diagnostic error, and improve patient outcomes, all while collecting a corpus of clean, labelled clinical data. Our system is currently deployed live at the Beth Israel Deaconess Medical Center Emergency Department and is in use by doctors.

Thesis Supervisor: David Sontag
Title: Associate Professor

Acknowledgments

The body of work I present here would not have been possible without guidance and support from so many. Chief among them is my advisor, Prof. David Sontag, who has been instrumental in my growth as a researcher. David has taught me too many things to include in this paragraph, but the most valuable of these will always be how to formulate the right questions, and when to conclude they are the wrong ones.

This project was truly a group effort, and I am grateful for my team. Dr. Steven Horng patiently offered his clinical expertise and persevered through hacky implementations of half-baked ideas, even in the midst of a pandemic. Luke Murray is the design wizard behind this project, and I'll miss the nights of frantic debugging and brainstorming. David Karger gave me crucial advice on user-centric design.

Thank you to Monica Agrawal for endlessly discussing every one of my musings and offering support and guidance with a smile (and a laugh). My MEng would not have been as productive or as fun without you and the rest of the Clinical ML group.

Thank you to my past UROP mentors and professors who gave me the building blocks and tools to become a researcher. Thank you to my family, for reminding me of the bigger picture and supporting me even at my worst. Thank you to my amazing and loyal friends, who have let me blabber on about random things and brought me more joy than I thought possible.

And lastly, thank you to MIT. My first impression of this school was one of pure wonder as I stepped onto Killian Court as a wide-eyed ten-year-old and saw buildings emblazoned with names of the scientific greats. While I cannot say that I have become the next Aristotle or Newton during my time at this institution, I am humbled by the people I have met and the opportunities I have been given. This school has changed my life, and for that, I am forever grateful.

Contents

1 Introduction **15**

1.1 Motivation 15

1.2 A New EHR 16

1.3 Thesis Scope and Organization 17

2 Background **19**

2.1 The Clinical Workflow 19

2.2 Related Work 21

2.2.1 Data Entry: Related Work 24

2.2.2 Contextual Information Retrieval: Related Work 26

2.3 Dataset Summary 30

3 Rethinking Clinical Documentation **33**

3.1 Data Entry and Contextual Autocomplete 35

3.2 The Sidebar 40

4 Generating Rankings for Autocomplete **43**

4.1 Developing a Clinical Language Model 43

4.1.1 Defining Clinical Concepts 45

4.1.2 Autocomplete Scope and Ranking 45

4.2 Defining the Autocomplete Problem 47

4.2.1 Label Generation 47

4.2.2 Featurizing Textual Data 49

4.2.3 Alternate NER Approaches . . . 50
4.3 Autocompletion Model by Concept Type . . . 52
4.3.1 Autocompleting Conditions . . . 53
4.3.2 Autocompleting Symptoms . . . 55
4.3.3 Autocompleting Labs and Medications . . . 58
4.4 Contextual Autocomplete Results . . . 58
4.4.1 Performance and Usability . . . 59
4.4.2 Sensitivity Analysis . . . 64
4.4.3 Interpreting Autocompletion of Prior Conditions . . . 66
4.5 Summary . . . 68
4.5.1 Future Direction: Dynamic Autocomplete Rankings . . . 70

5 Triggering Autocomplete Scope 71
5.1 Rule-Based Triggers . . . 72
5.2 Learned Triggers: The Setup . . . 73
5.2.1 Defining Labels . . . 74
5.2.2 Featurizing Text . . . 74
5.2.3 Dataset Generation . . . 75
5.3 Learned Triggers: Modeling . . . 77
5.3.1 How much local structure do we need? . . . 77
5.3.2 Binary Prediction of Autocomplete Scope . . . 78
5.3.3 Autocomplete Scope and Type Prediction . . . 83
5.4 Performance Results . . . 85
5.4.1 Scope and Type Detection . . . 85
5.4.2 Overall . . . 86
5.5 Summary . . . 88
5.5.1 Future Direction: Integrating Semantic Modifiers . . . 88

6 Patient Record Summarization from Unstructured Text 95
6.1 Formalizing OMR Snippetization . . . 96
6.2 Dividing Notes into Snippets . . . 98

6.3 Measuring Snippet Relevance 99
6.3.1 Advanced Keyword Search 99
6.3.2 Latent Dirichlet Allocation and Topic Modelling 104
6.3.3 Anchor-and-Learn 107
6.4 Deployment and Next Steps 113
6.4.1 Implementation of Advanced Keyword Search 113
6.4.2 Future Work 115

7 Information Synthesis and Visualization of Semi-Structured Data 117
7.1 Drug-Disease Indications 117
7.2 Lab and Vital Trend Visualization 118
7.3 Condition-Procedure Relations 119
7.3.1 Constructing an Ontology of Procedures 121
7.3.2 Establishing Prior Mentions of Procedures 122
7.3.3 Mapping Conditions to Procedures via Semi-Supervised Affine Transformations 123
7.3.4 Mapping Conditions to Procedures via Logistic Regression . . 125
7.4 Differential Diagnosis 127
7.4.1 Generalized Differential Diagnosis 128
7.4.2 Differential Diagnosis of Abdominal Pain 134
7.5 Future Work 140

8 Implementation & Deployment 143
8.1 System Implementation 143
8.1.1 Client-Side Inference 145
8.1.2 Ontology Modifications 145
8.1.3 Collaborative Documentation 146
8.2 System Feedback & User Metrics 147

9 Discussion and Conclusion 151
9.1 Limitations 151

9.2 Future Work 153

A Appendix **157**

A.1 NegEx Algorithm 157

A.2 Examples of Clinical Notes 157

A.3 OMR Annotation Tool 160

List of Figures

2-1 High-level summary of dataset demographics 31

3-1 Consequences of semi-structuring clinical notes 35
3-2 Screenshots of contextual autocompletion tool 36
3-3 Markdown formatting supported in our tool as a part of the data entry experience. 37
3-4 Slash commands for clinical concepts 38
3-5 Redundant entry is removed by including already tagged terms in early sections of notes to later ones. 39
3-6 Screenshot of sidebar cards 41
3-7 The end-to-end EHR system 42

4-1 Data model for our ontology of conditions 47
4-2 Backup data capture strategies when autocomplete scope fails 59
4-3 Mean MRR for conditions using contextual and frequency-based autocompletion 67
4-4 Case Studies of Autocomplete Rankings over Conditions 69

5-1 Precision-Recall curve for best learned scope/type prediction models . 87

6-1 OMR Snippetization UI 97

7-1 Screenshot of drug-disease indications in the system. 118
7-2 Box-and-whisker visualization of a lab trend. 119
7-3 Inserting aggregate lab/vital trends into the editor with autocomplete 120

7-4 Screenshot of condition-procedure relations in the system. 120
7-5 The effect of subsampling on diagnostic performance 134
7-6 Performance of multibranch-CNN to predict diagnosis for a chief complaint of abdominal pain, broken down by note length. 139

8-1 System Infrastructure 144

A-1 Pseudocode of the rule-based negation detection algorithm. 158
A-2 OMR Annotation Tool 161
A-3 Candidate OMR Snippets 162

List of Tables

2.1 Documentation workflow in the ED. 20
2.2 Sections of an ED physician note. 22
2.3 Data available in a patient's medical history. 23

4.1 Comparing NER approaches on OMR notes 51
4.2 Retrospective Evaluation of MRR using Contextual Autocomplete . . 62
4.3 Retrospective Evaluation of Keystroke Burden and MAP using Contextual Autocompletion 63
4.4 Live evaluation of contextual autocomplete ranking models 65
4.5 Keystrokes saved by contextual autocomplete by concept frequency . 65
4.6 Predictive features for contextual autocomplete condition ranking model (selected concepts) 66

5.1 Word2Vec embedding quality in learned autocomplete triggering . . . 76
5.2 Performance of binary autocomplete scope prediction models 82
5.3 Performance of autocomplete scope and type prediction models . . . 86
5.4 Autocomplete scope and type performance for the best-performing CNN model, broken down by class. 87

6.1 Case Studies: Snippets surfaced with standard vs. advanced keyword search for a sample of conditions and patients. 102
6.2 Case Studies: Snippets surfaced with standard vs. advanced keyword search for a sample of conditions and patients. 103
6.3 Visualization of LDA topics for OMR Snippetization 106

6.4 Visualization of LLDA topics for OMR Snippetization 108
6.5 Coefficients for Anchor-and-Learn Model for OMR Snippetization . . 111
6.6 Case Studies: Snippets surfaced with advanced keyword search vs. anchor-and-learn logistic regression for a sample condition. 112

7.1 Qualitative examples of closest procedures to conditions using affine transformations. 126
7.2 Performance of diagnostic models for general differential diagnosis . . 132
7.3 Highly correlated tokens for a selection of filters in the generalized differential diagnosis CNN model. 133
7.4 Diagnostic labels for chief complaint of abdominal pain or related. Labels are ordered by specificity, such that if a visit is assigned a ICD code from two or more rows, it is labeled with the more specific diagnosis. N represents the number of samples in the given class. 136
7.5 Performance of diagnostic models for a chief complaint related to `abdominal pain` 139

A.1 Hyperparameters for tuning deep models for differential diagnosis . . 161

Chapter 1

Introduction

The advent of electronic health records (EHRs) in recent years has transformed medicine. In this thesis, we present a new EHR system that reduces cognitive load on doctors by leveraging machine learning to decrease documentation burden and provide clinical decision support. In this chapter, we outline the structure of this thesis and argue why we must reform EHR systems.

1.1 Motivation

Clinicians currently spend more time documenting information in EHRs than communicating with patients, and the timesink in using inefficient EHRs is posited to be a leading cause of physician stress and burnout [18, 39]. Doctors prefer using natural language and free-text for documentation over restrictive structured forms [58], but clinicians have adapted to time-intensive note-writing by relying on overloaded acronyms and jargon [110].

Consequently, medical documentation is often noisy, ambiguous, and incomplete. The lack of structure in notes further hinders comprehensibility for patients, other physicians, and machines, and makes patient handoffs cumbersome [6, 41, 59]. The information within EHR notes remains largely untapped and, at present, cannot be easily

accessed for downstream medical care or for machine learning models that rely on structured data. Recent advances in supervised machine learning were largely enabled by clean, labeled datasets containing millions of datapoints such as ImageNet and CIFAR-10. The lack of sufficiently labeled data poses a challenge, and clinical data cannot always be retrospectively annotated in a reliable way. In addition, widespread variation in EHR documentation interferes with patient care by making clinical notes unintelligible [27] or even leading to errors in prescribing medications [60, 11].

There is a clear and present need for better EHR interfaces that aid physicians. While new EHR software programs are regularly created and deployed, they continue to be a cumbersome necessity for physicians and have yet to help improve patient outcomes.

1.2 A New EHR

In this thesis, we present a completely redesigned EHR system for Emergency Department (ED) physicians. This EHR uses an ML-powered intelligent user interface for physicians to employ in realistic clinical documentation settings. The goal of this project is two-fold: first, to bring machine learning research closer to medicine by surfacing intelligent suggestions that might aid a physician in making a diagnosis or documenting a patient's medical record; and second, to easily provide physicians with the ability to create structured annotations and labels for clinical data. A core part of our design is to reward users for creating meaningful annotations by enabling downstream clinical decision support with tagged terms. In essence, creating clinical labels is inherently useful to a doctor even at the point-of-care, which we demonstrate with this tool.

Our EHR tool consists of two parts:

1. *Intelligent data entry*, wherein a physician can easily tag and insert clinical concepts with an autocomplete functionality. We provide learned predictions

over clinical concepts using the medical context of the patient.

2. *Contextual information retrieval*, wherein the tagged concepts then immediately enable clinical decision support. Physicians can easily explore disease history, visualize laboratory trends, and peruse related data.

1.3 Thesis Scope and Organization

The long-term goal of this project is to completely redesign commercial EHRs, which will in turn reduce physician burnout, facilitate more rapid processing of clinical data, decrease diagnostic errors, and improve patient outcomes. It will also result in a large corpus of high-quality, structured clinical data. This project is a collaboration between MIT and Beth Israel Deaconess Medical Center (BIDMC) and builds on an eight-year collaboration between my PI and advisor, Prof. David Sontag, and the BIDMC Co-PI, Dr. Steven Horng. Dr. Horng is an attending physician within the BIDMC ED. BIDMC already has a custom-built EHR that is used by all of its staff, and we developed our EHR utility on top of this stack. This allowed for rapid prototyping and quick iteration, as well as access to anonymized EHR data that was used to train machine learning models for the live tool. Our system is currently deployed live within the BIDMC ED, and is used by Dr. Horng and others for clinical documentation.

This undertaking required significant expertise in both machine learning and human-computer interaction (HCI). As such, we worked in conjunction with Luke Murray, who is a PhD student within Prof. David Karger's Haystack group. This thesis touches upon the complete design of our tool but specifically focuses on the machine learning aspects that drive it, and does not detail many of the specific UI choices that are in the final interface.

Finally, this project and ongoing collaboration is funded in part by a grant from MIT's Abdul Latif Jameel Clinic for Machine Learning in Health (J-Clinic). In this thesis, we outline the initial pilot of our tool, but note that it is constantly evolving.

Thesis Organization

We first describe the overall clinical workflow and necessary background information in Chapter 2. Chapter 3 provides an overview on the design of the tool and its constituent features. Chapters 4 and 5 detail the system's intelligent data entry mechanisms via contextual autocomplete. Chapters 6 and 7 discuss contextual information retrieval and clinical decision support for the tool, via unstructured retrieval of text snippets (Chapter 6) and information synthesis and data visualization of structured data (Chapter 7). In Chapter 8, we detail the implementation and deployment of the tool from a systems perspective as well as our strategy to collect metrics and other sources of feedback from the tool. Finally, in Chapter 9, we conclude with a reflection of the overall tool as well as potential future work.

Chapter 2

Background

2.1 The Clinical Workflow

Once a patient enters the ED, there are several points of interaction with clinical staff. These are summarized in Table 2.1. The patient first meets a triage nurse who records patient vitals as well as a short description of why the patient decided to come to the ED. This triage note is summarized in a succinct one- to two-word phrase known as the Chief Complaint. After the patient is admitted to the ED, doctors and nurses each maintain notes which are updated throughout the course of the visit and contain information about the patient's history, current presentation, pertinent labs and tests, and a final diagnosis and treatment plan. Composing these unstructured notes is a time-intensive process; our work focuses on decreasing documentation burden within the doctor's note. This note is eventually converted to an official discharge summary that is filed in the patient's record. It is thus a constantly evolving document, and is edited before the doctor sees the patient (to document patient history), while treating a patient (to document relevant symptoms and tests), and after the patient is discharged (to document the final diagnosis).

The doctor's note itself (and the subsequent discharge summary) are completely free-text. However, modern versions of these notes in the BIDMC ED do have section

Emergency Department Documentation Workflow			
	Documentation Type	*Documentation Description*	*Example*
1	Triage Assessment	Short note describing patient's status (free-text), vitals (blood pressure, temp, pulse ox, heart rate, respiratory rate)	`patient complains of chest pain s/p fall. has trouble breathing.`
2	Chief Complaint	One-phrase summary describing reason for visit (structured)	`CHEST PAIN`
3	Nurse notes	RN comments describing patient and care throughout visit (free-text)	`36 y/o female with CP, no prior cardiac history`
4	Doctor notes	More thorough MD comments about patient including history, current status, diagnosis, and treatment (free-text)	`36 y/o F p/w CP s/p fall, no prior cardiac history, family history of ...`
5	Discharge summary	Official note filed in medical record that summarizes visit, usually updated from MD comments (free-text)	`Patient is a healthy 36 year-old woman who came in complaining of...`

Table 2.1: Documentation workflow in the ED.

headers to delineate different parts of the patient's demeanor. For the most part, doctors edit sections sequentially, as the first few sections correspond to patient history that can be gathered prior to seeing the patient, while the later sections delve into lab and test results that affect the patient's current diagnosis. Each section and its purpose are listed in Table 2.2.

Clinical staff can also access the patient's medical history, which is a rich resource. Elements of the online medical record (OMR) are listed in Table 2.3. The bulk of the information in the OMR lies within older clinical notes in the patient's file, and these notes contain detailed descriptions about disease history and prior clinical care. Unfortunately, the OMR is long and difficult to quickly parse– in our dataset, the median number of OMR notes per person is 34 with a median note length of 301 words.

2.2 Related Work

Over 96% of reported hospitals in the U.S. used an EHR technology in 2015, yet the ubiquity of EHRs has not helped physicians. EHRs pose a huge timesink and cause physician stress and burnout which in turn limits patient interaction [31, 39]. Many EHR systems lack support for typical clinical tasks such as decision making and review of a patient's treatment chart, and don't allow the clinician to easily retrieve patient data [55]. Clinicians thus have to resort to writing notes in unstructured text boxes, using overloaded jargon and ambiguous terminology to alleviate documentation burden [110]. While this abbreviated clinical language may be understandable to the note author, it can be unreadable to doctors in other specialties, making patient handoffs tricky [37]. Moreover, these notes are hard to retrospectively annotate, creating a dearth of labeled clinical text for use in the machine learning community.

The potential of EHRs to transform clinical documentation and the medical practice has been acknowledged since their inception. Over twenty years ago, [114] proposed an EHR workflow wherein disparate sources such as medical literature, knowledge

Sections of ED Physician's Note			
	Section	*Description*	*Example*
1	History of Present Illness (HPI)	Short summary of patient's chief complaint and core symptoms, with any relevant prior conditions.	`27 y/o M p/w chest pain w/ h/o congestive heart failure...`
2	Past Medical History (PMH)	A list-based summarization of the patient's full medical history, regardless of whether it is directly relevant to the chief complaint.	`CHF, heart surgery in 2004, toe amputation, ...`
3	Medications	A list-based summarization of the patient's current medications. This can sometimes be pre-filled using structured data.	`Metoprolol, metformin, coumadin (for afib), ...`
4	Family History	Explanation of any conditions that might have a genetic component or predisposition, and which relative has it.	`Diabetes in mother, father hypertensive. No h/o cancer.`
5	Social History	Short description of patient's eating/-drinking/drug habits.	`No smoking or drugs. Drinks alcohol socially.`
6	Review of Systems (ROS)	A question/answer form that maps patient symptoms to general organ systems (neurological, pulmonary, constitutional, etc.)	`Constitutional: nausea, fever. Psych: auditory hallucinations.`
7	Physical Exam (PE)	A form mapping areas of the body to any noticeable signs/symptoms found during a physical exam of the patient.	`Abdomen: tender in right upper quadrant, Pupils: dilated`
7	Medical Decision Making (MDM)	Explanation of how the physician made the final diagnosis, including restatement of symptoms and past conditions, tests ordered, relevant lab values, and proposed course of treatment.	`39 y/o p/w arm pain, ordered x-ray, showed wrist fracture...`
8	Diagnosis (Dx)	Short phrase with final diagnosis.	`Wrist fracture`

Table 2.2: Sections of an ED physician note.

Online Medical Record (medical history)		
Data Source	Data Type	Example
Medication history	Structured	`Coumadin 2mg tablet (prescribed 1/27/2006), ...`
Lab test history	Structured	`2/4/2004: (HCT, 44), (CREATININE, 1.2)`
ICD Diagnosis Codes	Structured	`2/4/2004: 250.0, 530.81`
Problem list	Semi-structured	`[Hypertension, type 2 diabetes, afib...]`
Surgical history	Semi-structured	`[Appendectomy (6/5/2007), coronary artery bypass (8/19/2009), ...]`
Medical notes	Unstructured	`Patient visited the ED on...`

Table 2.3: Data available in a patient's medical history.

bases, patient-specific information, and clinical algorithms could be joined. Clinical cognition tools that use machine learning to encode medical expertise have been around since the 1970s [86]. Yet, early research in medical informatics and patient care management is almost divorced from practice [36]; modern EHR systems are known to have a multitude of usability and interface design issues, which can even lead to documentation mistakes [63]. In a Finnish study examining user satisfaction for commercial EHRs, physicians feared that EHR systems were developed by engineers and doctors in administrative positions rather than practicing clinicians, and this resulted in tools that were not immediately helpful [76]. While study participants acknowledged that EHRs offer electronic checklists, the ability to connect medical records from multiple organizations, and faster ways of storing and loading data, they also had a long litany of complaints. Clinicians wanted easy statistical aggregates of quantitative information, as well as a dynamic system that "predicted users' movements." They also asked for variable views of a patient with both high-level summaries and the ability to drill-down to specific details, effective searching through the patient's prior notes and labs, and decision support. In addition, system

interfaces used "engineer-language" and seemed unintuitive, including multiple clicks for simple tasks that actually increased documentation time. One participant voiced a request for the better use of color and graphical icons to indicate components of the system [42].

Clinicians also found existing EHR solutions unsuitable for the collaborative nature of medical treatment [55]. Even at a single institution, users found it difficult to convey information between doctors and other clinicians. The commercial EHR landscape is also notoriously fragmented, indicating a need for better system interoperability with other medical tools and software vendors [117] – the lack of consistency means that EHRs do not support achieving continuity of care. Finally, clinicians in one study noted that performing simple information retrieval or data exploration tasks often required interrupting their documentation workflow [56]. The suggestions made by physicians indicate that a successful EHR must be a live, collaborative document that allows for repeated input, output, and iteration from multiple parties in a seamless and understandable way. [36] summarizes this succinctly: EHRs should get "the right information to the right place at the right time."

2.2.1 Data Entry: Related Work

There have been attempts to mitigate the EHR information overload by creating semi-structured representations of a patient's medical history such as the problem list, which catalogs a patient's prior diseases and conditions. However, these lists are poorly maintained, incomplete, and inconsistent amongst practitioners [100, 119]. [36] cites the lack of a standardized clinical vocabulary as a core challenge in medical informatics. Efforts to structure free-text within clinical notes have been limited and fail to utilize machine learning to understand the current medical context. One common technique is for providers to generate auto-populated templates with boilerplate text and previously captured patient data [122]–for example, clinical notes usually begin with information about patient demographics and the chief complaint, like `"26 y/o M complains of dyspnea"`. Of course, this method only works for a few

small sections of notes, and it does not generalize well across practices or capture subtleties of documentation such as physician preference. [118] proposes creating a semi-structured clinical record by forcing doctors to document clinical concepts via structured lists that then can be transformed to free-text using text generation algorithms; they contend, however, that the notes created with the system are hard to parse and that clinicians find it slow to enter information in this restrictive manner. [54] creates semi-structured medical records using XML tags to denote sections of the note, billing codes, and patient demographics, and suggest using natural language processing (NLP) to retroactively detect clinical concepts once the note is written. This post-hoc way of identifying concepts does not solve underlying issues with concept disambiguation.

In addition, crafting ontologies of clinical concepts that doctors can easily use in a live interface is difficult. The desiderata for clinical interface ontologies differ from those of other healthcare ontologies used for data storage (e.g., SNOMED-CT), information retrieval (e.g., MeSH), and classification (e.g. ICD codes) [101]. Interface ontologies need to balance between pre- and post-coordination of clinical concepts, encode levels of concept specificity, be easy to visually navigate, and contain domain-specific jargon.

To date, the most successful attempt to ease documentation burden in the ED with machine learning is by Greenbaum *et. al* [47], who construct a model to predict candidate chief complaints from a set of around 200 standardized phrases. The model – a multiclass SVM trained on triage information – is able to structure around 99% of chief complaints in the authors' ED. However, this work does not address greater pain points in the documentation process such as surfacing relevant patient history and noting down present symptoms. We innovate on the work in [47] to provide contextual autocomplete functionality for the entire clinical workflow by architecting a model that incorporates both present information (triage text, vitals) and past medical history (OMR), and by building an interface that supports the straightforward documentation of multiple terms from a large space of possibilities.

Of course, there are successful elements from other clinical documentation utilities

that we incorporate into our tool. [106], for example, shows that easy navigation of forms with fixed text-boxes for different categories of information can allow for the collection of standardized data much faster than raw free-text can. In addition, [103] creates a clinical documentation tool for ophthalmologists centered around end-user programs, where physicians can paste customized templates automatically populated with patient information into notes. We implement versions of both of these features, dubbed *tabbed forms* and *macros* respectively, and detail them in the next chapter. Previous implementations of customizable templating of clinical documents can potentially hinder note shareability when templates differ between clinicians and institutions [120], but our system maps all clinical concepts to publicly available medical ontologies, which combats this issue.

2.2.2 Contextual Information Retrieval: Related Work

While easy clinical documentation remains a relatively unsolved problem, there has been a concerted effort to use machine learning to enable clinical decision support via information retrieval and data visualization. An early example of this is the HARVEST system, which is a patient record summarization tool developed by Columbia University and New York-Presbyterian Hospital [34]. HARVEST is comprised of three core features: (1) a timeline that lists past dates of admission as well as the resulting diagnosis from each visit; (2) a problem cloud that shows the most commonly mentioned terms in a patient's OMR by frequency; and (3) a notes panel that allows doctors to view and search through past OMR notes using an exact string match. In the subsequent HARVEST evaluation paper, the authors note that the tool is most useful to abstract metrics from patients with long hospitalizations via the patient timeline, and that the problem cloud, while a nice visual representation, was not immediately useful to clinicians [90]. Moreover, the OMR search tool allowed doctors to parse through individual notes, but did not point users to particular paragraphs of interest based on the clinical context.

HARVEST and other contemporary EHR systems fundamentally provide data orga-

nization but little to no information synthesis [65]. In addition, chronically ill patients tend to have denser clinical records, making patient summarization uniquely difficult for critical patients. In one study, 37% of surveyed general practitioners reported that they sometimes gave up searching for information in a patient's medical record because it was too time-consuming [24].

Patient Record Summarization

Intelligent patient summaries, if baked into a simple user interface, could be incredibly useful to physicians by obviating the need to manually search through a large corpus of notes for relevant information. While there have been a myriad of algorithms that summarize text well in the general domain (e.g. for news and scientific articles), clinical text summarization has yet to see this success [7]. This is largely because of the inherent messiness of clinical text, including information redundancy, complex temporality, missing data, salience detection, and barriers to deployment [91].

There are two dimensions along which to categorize summarization algorithms. The first describes how much summaries rely on the original input text. An *extractive* summary that borrows exact phrases or sentences from a corpus to form a synopsis about the set of documents. On the other hand, an *abstractive* summary generates new text to synthesize information in the input text [91]. Clinical text summarization has focused mainly on the former, as the latter requires a well-formed model of clinical natural language generation. All of the abstractive clinical summarization systems reviewed in [91] rely on extensive rules and hand-crafted medical ontologies for very specific clinical specialties [98, 77, 51]. None are deployed in live environments. Recent efforts in extractive summarizations of clinical text have been more fruitful, but are still limited to specific diseases [68], semi-structured note types [9], or to extracting note-level topics and themes rather than sentences of interest [45, 111]. In the ED, physicians must instead browse through a multitude of note-types and potential prior conditions.

In addition, the way in which a summarization tool is intended to be used can affect

its categorization. An *indicative* summary points to and highlights portions of the original text to the reader (e.g. relevant lab values), and are meant to be used in conjunction with a patient's full medical record. In contrast, an *informative* summary is used as a standalone synthesis of patient information that is designed to completely replace the original data [91]. Clinical text summarization is not yet in a place where it can truly be used as an alternative to reading through a full medical record, so we instead seek to create *extractive* and *indicative* summarizations of a patient's medical history by using salient snippets from a patient's OMR. In doing so, we reduce documentation burden by synthesizing and displaying information in one interface.

Information synthesis for ED physicians is also distinctive because it is not inherently longitudinal. If a diabetic patient enters the ED, a doctor does not care about their entire medication history but rather the current treatment regime and how the patient feels at present. ED physicians are thus less concerned with building a picture of a patient's care over long timescales. Most clinical text summarization tools operate on a per-note basis, but this is still too granular for an ED physician to utilize. We instead wish to capture some notion of *relevance* in our extractive summarization by finding key snippets that can succinctly capture the patient's current state of care while excluding out-of-date information.

Extractive summarization efforts have also focused on finding snippets of published medical literature that relate to a patient's current condition [32, 33]. Furthermore, relevance cannot simply be measured by co-occurrence of terms. For example, a document discussing unstable angina in male patients would not be relevant to a female patient, despite a close similarity in thematic content [33]. As such, it is often necessary to use a combination of manual and template-based labelling to determine the context of different medical terms within a text, and to use these manual annotations to more deterministically establish relevance. Finally, iterating on any summarization tool using feedback from end-user clinicians is absolutely necessary, as the prioritization between the succinctness of summaries, relevance to the specific patient at hand, and ease of parsing the output summary text is nontrivial and dependent on different

clinical use-cases. In our work, we extract relevant information from previous clinical notes associated with a patient, not generic journal articles. While we operate in a patient-specific domain, we can still use several of these principles for detecting relevance in potentially contextual documents.

Structured Information Synthesis and Visualization

There is also a wealth of structured information in medical records that can be equally helpful to contextualize and diagnose a patient. This includes medication, lab, and vital signs, as well as past procedures. In order for BIDMC clinicians to browse through this data at present, they are forced to click out of the note that they are drafting and comb through pages of raw data. We instead propose building a single EHR interface that can easily synthesize and display historical data without breaking the clinical workflow.

Unfortunately, data visualization in healthcare is severely behind other fields [16]. Many older clinical visualization systems split information across multiple screens and tables, making it difficult to parse [8]. In one study designed to test interactive systems for a lung transplant home monitoring system, clinician readers preferred graphical, interactive displays of data rather than in the standard tabular form, even if the data was temporal and/or multivariate [89]. These tools also tend to support either a high-level exploratory analysis task or a specific low-level query [121] – this leaves a gap for a tool that can support intuitive hierarchical data exploration. As an example, in response to a patient's most recent lab value, a doctor might want to know whether it is abnormal compared to the patient's baseline, or perhaps explore the historical lab trend in greater depth with a line chart. There also exists a tradeoff between specificity and generalizability of visualization systems for different data types. While limiting the number of visualization types to a few helps users familiarize themselves with the system, finding a universal representation of data across differing data types (lab values, medication dosages, vital signs) is impossible [97, 93]. There is a need to construct a canonical process model of clinical data visualization – one

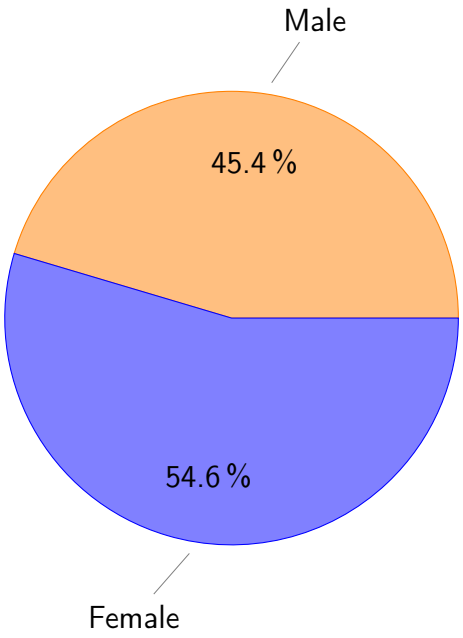
that incorporates hierarchical data visualizations with graphics rather than tables, while standardizing the types of visualization techniques so as not to overwhelm the user. [113] proposed a taxonomy to medical data visualization over two decades ago, yet we still have not standardized how to present clinical information.

[70] also presents a broad manifesto of how artificial intelligence can transform clinical documentation through voice-to-text note scribing, information synthesis, patient risk stratification, and differential diagnosis support. We touch upon many of these suggestions in subsequent chapters.

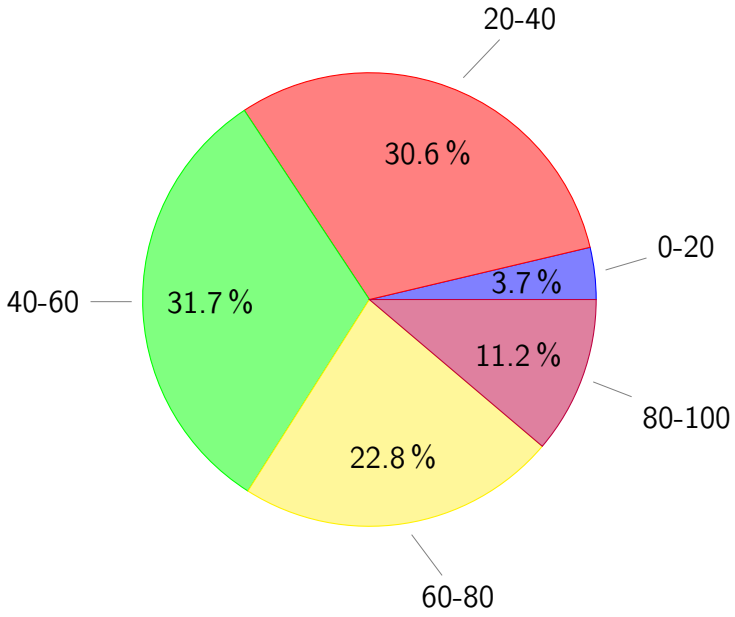
2.3 Dataset Summary

We use data from 273,000 de-identified visits to the BIDMC ED over the last decade, representing around 140,000 unique patients. For each visit, we have access to patient demographics, triage information (triage assessment, vital signs, and chief complaint), clinical notes from both the doctor and nurse assigned to the patient, and currently prescribed medications. The clinical notes do not represent the final discharge summary that is filed in the patient's OMR, but rather a draft copy of the doctor's notes known as the `MDComment`. As such, the `MDComments` do not always have the section headers that appear in final copies of notes as detailed in Table 2.2, nor are they necessarily complete. The noisiness of these notes poses interesting challenges in terms of how to frame any of our learning problems, which we discuss in future sections. A high-level summary of patient demographics is shown in Figure 2-1.

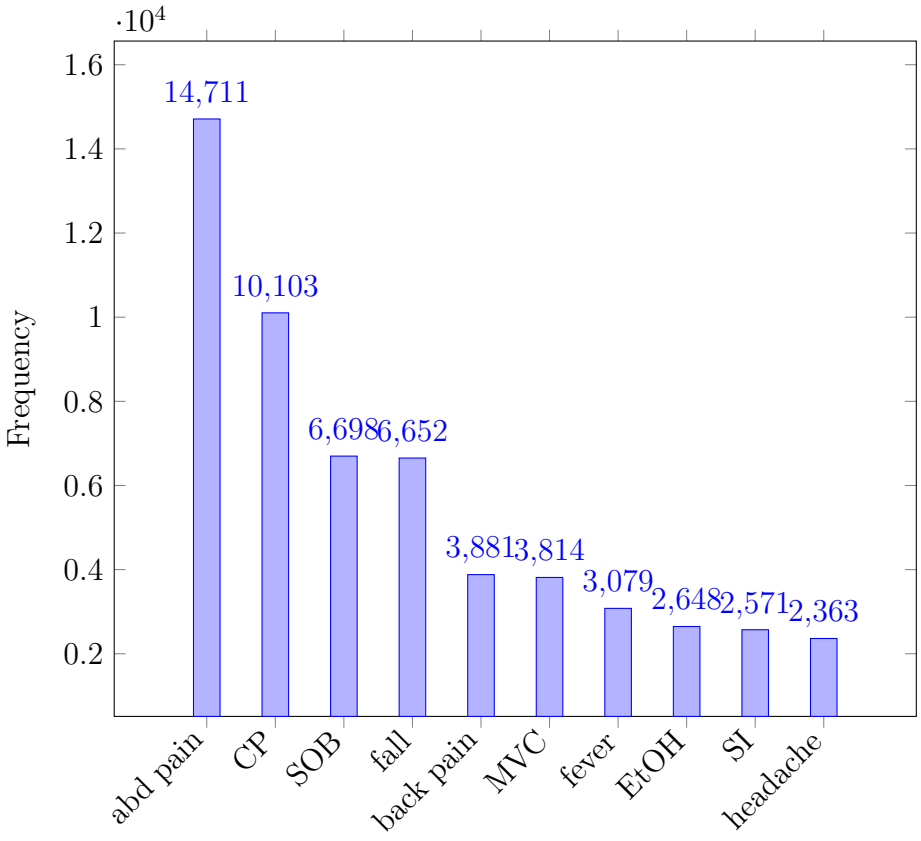
In addition, we have all prior OMR notes for patients who had previously been in the hospital system (74% of visits). We do not restrict our analyses to any particular subset of these visits as we seek to build an all-encompassing system that serves patients both with and without extensive medical histories on file.



(a) ED Notes by Sex



(b) ED Notes by Age



(c) Top Chief Complaints by Frequency

Figure 2-1: High-level summary of dataset demographics

Chapter 3

Rethinking Clinical Documentation

In this chapter, we outline the core parts of our ML-driven EHR utility, and discuss high-level system design and implementation. In later chapters, we detail the ML algorithms that are the backbone of the tool as well as specific system implementation and deployment information.

Our system is comprised of two elements: (1) *intelligent data entry*, where we allow a physician to tag and insert clinical concepts with an autocomplete functionality; and (2) downstream *contextual information retrieval* and clinical decision support, which uses tagged concepts to enable information synthesis and data exploration. Our primary goal in building a new EHR is to facilitate rapid capture of structured clinical concepts at the point-of-care via learned suggestions. The consequences of doing so are threefold:

For doctors: When a doctor tags phrases in a note, these terms uniquely identify clinical concepts and can be linked with relevant information from the medical record. Moreover, these terms can facilitate widespread improvements in documentation and reduce overall cognitive load on doctors. Once a term is tagged, it can be automatically inserted in multiple locations within the clinical note to limit the amount of redundant information a user types – for example, a tagged symptom in an earlier part of a note can be automatically appended to the Review of Systems section that

appears later on. This mitigates the “death by a thousand clicks" phenomenon that EHRs suffer from [107]. Live-tagging clinical concepts can also provide immediate rewards to physicians in the form of decision support. The captured structured data can then be used to build smarter EHR interfaces that enable contextual information retrieval about disease history and lab trends, without ever leaving the note interface.

For algorithms: The efficacy of machine learning and NLP in particular has thus far been limited in the clinical domain because of the lack of clean, structured training data [112]. Tagging clinical concepts automatically imposes structure on previously free-text notes. Prior research in clinical concept extraction tries to retrospectively recover concepts from notes, but these methods often struggle to disambiguate between similar concepts, and suffer due to a lack of labeled data [74]. Even for an expert physician, it is difficult to reliably disambiguate between concepts when the clinical intention is unclear. Building a tool that allows clinicians to document terms on the fly not only decreases documentation burden, but also curates large-scale prospective datasets of labeled clinical concepts (e.g. conditions, symptoms, labs, and medications) in notes. These labels can be used to design robust medical knowledge graphs, develop better clinical entity extraction models, learn longitudinal patterns within disease history, and even build contextual representations of concepts. Learning relationships between clinical concepts, for example, can inform and fill gaps in existing medical ontologies [21].

For patients and downstream medical care: Tagging clinical concepts with our system allows for the translation of acronyms and domain-specific language to common names. By normalizing key clinical concepts from notes to a common data model such as UMLS, notes become semi-structured and parseable. As an example, consider a real clinical note from the ED: `pt w/ h/o MS`. While `MS` might represent `mitral stenosis` to a cardiologist, it also can be used to denote `multiple sclerosis` to other specialists. To a layperson, the clinical note may be incomprehensible unless acronyms are expanded: `patient with a history of`

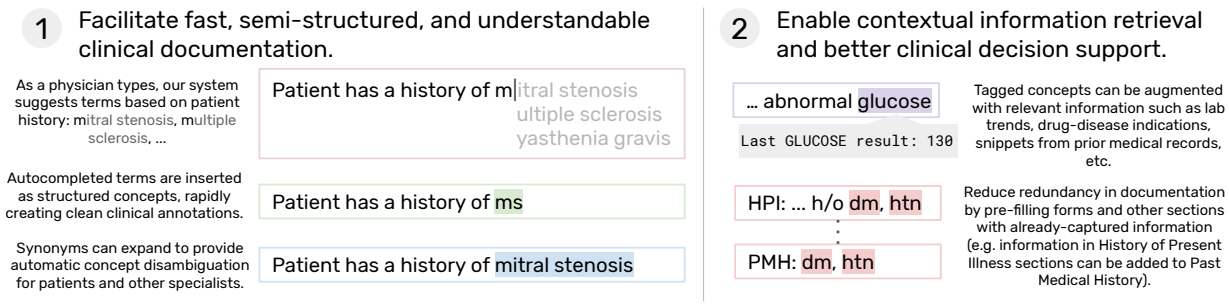


Figure 3-1: Consequences of semi-structuring clinical notes
Semi-structuring notes with tagged terms can enable extensive changes to both documentation and clinical decision support.

`multiple sclerosis`. Tagging `MS` at the point-of-care removes disambiguation while allowing for acronym and synonym translation.

We present intelligent data entry of structured concepts as the cornerstone of an new EHR in Figure 3-1.

3.1 Data Entry and Contextual Autocomplete

The workhorse of the data entry system is contextual autocomplete. This allows a user to tag clinical concepts of varying types via a deconstructed language model that predicts terms to document from historical and contemporaneous clinical concepts. Paradigmatic screenshots of the tool are shown in Figure 3-2. Prior medical records, current triage information, and the words that the clinician types all factor into our contextual autocomplete model, which we describe in detail in Chapters 4 and 5.

Tagged terms are normalized to UMLS and then are inserted as *cards* on the sidebar, which houses all of the contextual information retrieval and auxiliary data associated with a given concept. Clinicians can explore the tagged concept in further detail via these cards. Concepts can also be tagged with a manual trigger that shows the autocomplete dropdown when our predictive algorithms fail, as well as a postcorrection mechanism that automatically tags terms that can be easily disambiguated after it is entered, as described next in Chapter 4.

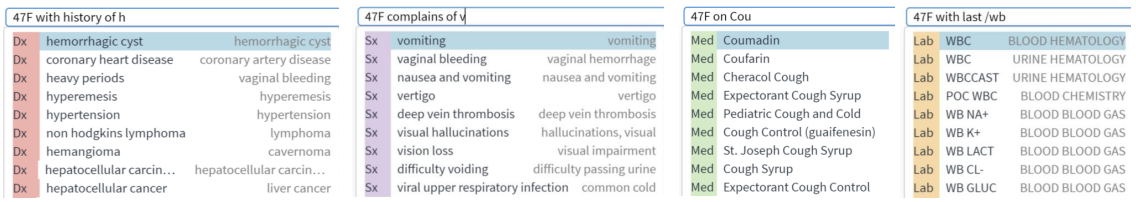


Figure 3-2: Screenshots of contextual autocompletion tool for each autocompletion type. From left to right: (a) Conditions (b) Symptoms (c) Medications and (d) Labs. Trigger words before the tagged term affect the scope and type of the autocompletion. Clinical concepts with synonyms that match the typed text are listed with the synonym in black text and the more general concept name in gray.

Of course, we seek to reward doctors for their annotations. While the sidebar is one mechanism of doing so, we also provide a few other features to ease the data entry experience: Markdown formatting, slash commands, removal of redundant data entry, macros, and tabbed forms.

Markdown formatting: First, we support common HTML/Markdown formatting to better organize information: unordered, ordered, and checkbox lists. These can be triggered with intuitive characters: typing `-`, `1.`, and `[]` at the beginning of a line to create a bulletted, numerical, and checkbox list respectively. A screenshot of this is shown in Figure 3-3.We have found that doctors find the checklists especially useful to keep track of tasks, such as ordering labs or asking for a consult from another service. This follows from previous studies in the ICU indicating that the use of checklists can improve patient outcomes [92].

Slash commands: We provide slash commands in the interface, which allows the clinician to filter autocomplete results to only a single concept type. Doctors can also use slash commands to insert rare clinical concept types, such as vital signs. Vital signs are rarely recorded in clinical notes, and are hard to retrospectively detect– consider the phrase `fever (101)`, where `101` refers to the patient's temperature; this would require some domain knowledge to understand. Vitals are structured information and are thus easy to pull from the patient's record and document. A

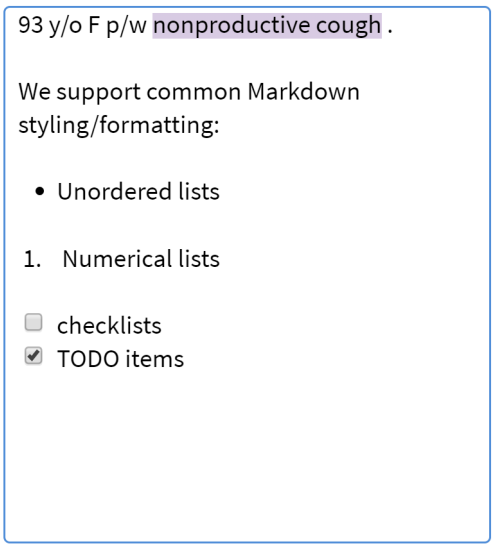


Figure 3-3: Markdown formatting supported in our tool as a part of the data entry experience.

screenshot of the slash command data capture for vitals is shown in Figure 3-4.

Removal of redundant data entry: Empirical studies of EHR notes entered by clinicians as free text have revealed that much of the information content of a note can be found in older documentation. Towards the end of a patient's stay in a hospital, around 70% of information in new text entries was found to have been copied from other notes [123]. Redundancies decrease readability and encourage the propagation of errors, and by automatically collecting and syncing redundant data we can mitigate these issues while allowing clinicians to continue to enter information in a natural way. Thus, we reduce redundant entry by automatically entering and updating information that is tagged in previous sections of the notes for later sections in three ways:

- Redundant entry of medication data: Medications tagged in early sections of the note are included in the `MedRecon` section later on. The medications are formatted in a list.
- Redundant entry of past medical history: Conditions tagged in early sections of the note are included in the `PMH` section later on. PMH is also stored in a textual representation of a list.

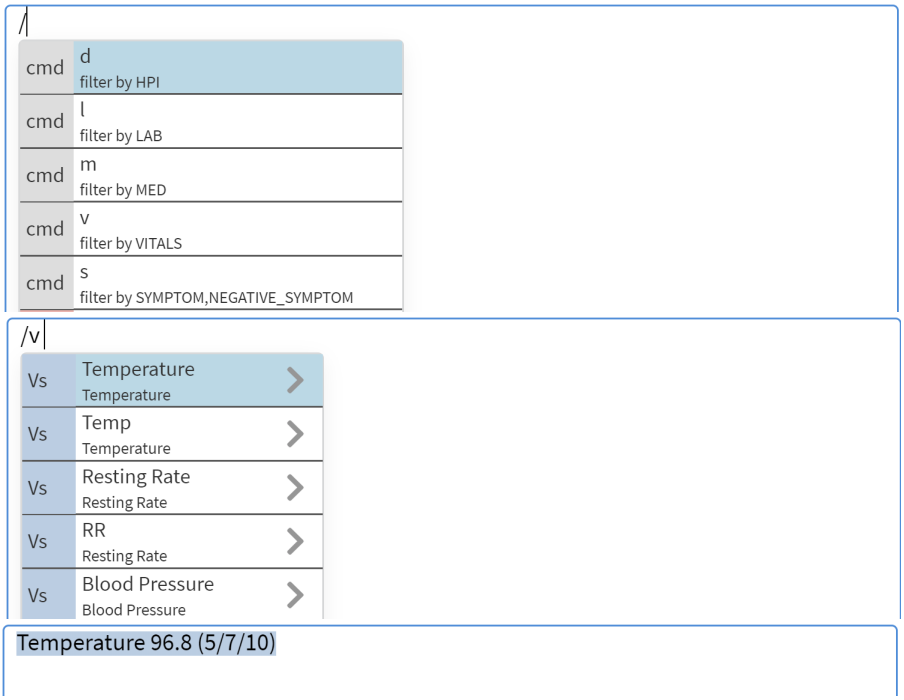


Figure 3-4: Slash commands for clinical concepts (conditions, symptoms, diseases, labs, and vitals) are shown here. Although vitals are not shown in the autocomplete dropdown because they are not inserted into the note often, they can be pulled from structured records and inserted into the note via *slash commands*. In the first image, users trigger the slash commands with the / character. Then, a dropdown of vitals is shown along with the timeframe the user wishes to aggregate information. Finally, the desired text is inserted as a concept.

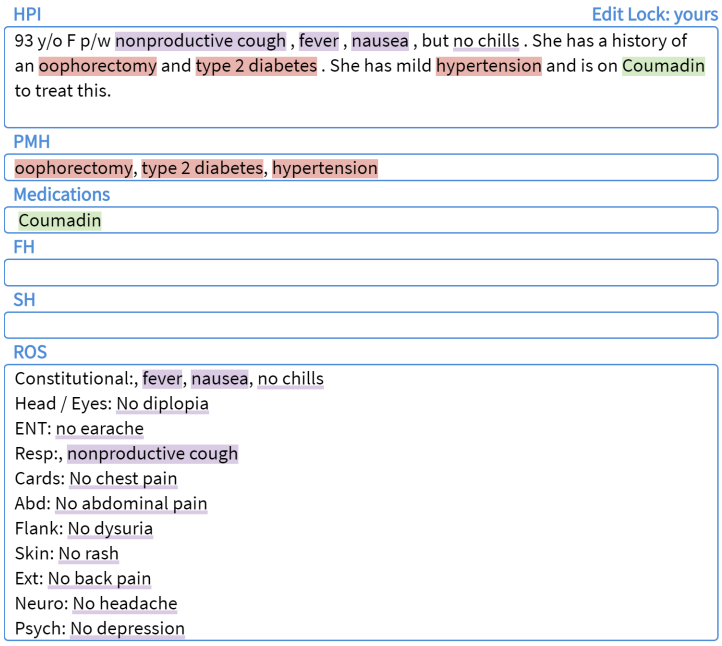


Figure 3-5: Redundant entry is removed by including already tagged terms in early sections of notes to later ones.

- Redundant entry of symptoms: Symptoms tagged in early sections of the notes are updated in the `Review of Systems` section later on. The `ROS` section is essentially a form which list symptoms corresponding to different body systems. Each symptom in our ontology was manually mapped to one of these body symptoms and is inserted in the appropriate place.

A screenshot of the redundant data entry can be seen in Figure 3-5.

Macros: Research from the HCI literature suggests that doctors can use programmatic note templates to reduce boilerplate text entry [103]. We adopt a similar form of end-user programming via the introduction of *macros*. Macros are function declarations (denoted by the `@` symbol) that can automatically add user-defined plaintext to the note. A macro like `@alcoholworkup`, for example, might insert a checklist with tasks for the doctor to complete such as lab orders.

Tabbed Forms: Finally, while not officially supported yet, we have the infrastructure to support *tabbed forms*. As can be seen in the `ROS` section of Figure 3-5, a lot

of the clinical note consists of sequences of questions of the form `FIELD : ANSWER`. Users can easily tab through these forms to enter information quickly. This is inspired by early literature on clinical decision/support systems, which suggest improving how doctors navigate text-based forms [106].

3.2 The Sidebar

The sidebar acts as a platform to facilitate data exploration and information retrieval. As concepts are tagged in the main editor of the note, they are also added to the sidebar via *cards*, which aggregate relevant information about the given concept and surface potential connections that the doctor can explore. An example of these cards are shown in Figure 3-6. Cards appear automatically as the user tags terms in the tool, and can be added and removed at will.

There are five main sources of data exploration and information retrieval that are currently supported by the sidebar at the time of writing:

- Related medications to a condition, using collected drug-disease indications.
- Visualizations of numerical data such as labs and vitals, using the windowed box-and-whisker plots described in Section 7.2.
- Related labs and vitals to certain symptoms, using hard-coded relations provided by physicians. As an example, knowing `CD4` and viral load lab values is important for a patient who has a tagged history of `HIV/AIDS`.
- Related procedures to a condition, using the learned relations described in Section 7.3.
- Related snippets to both conditions and medications, using procedures described in Chapter 6.

The first three rely completely on structured data (labs, vitals, and medications). The latter two, however, require pulling information from unstructured text data.

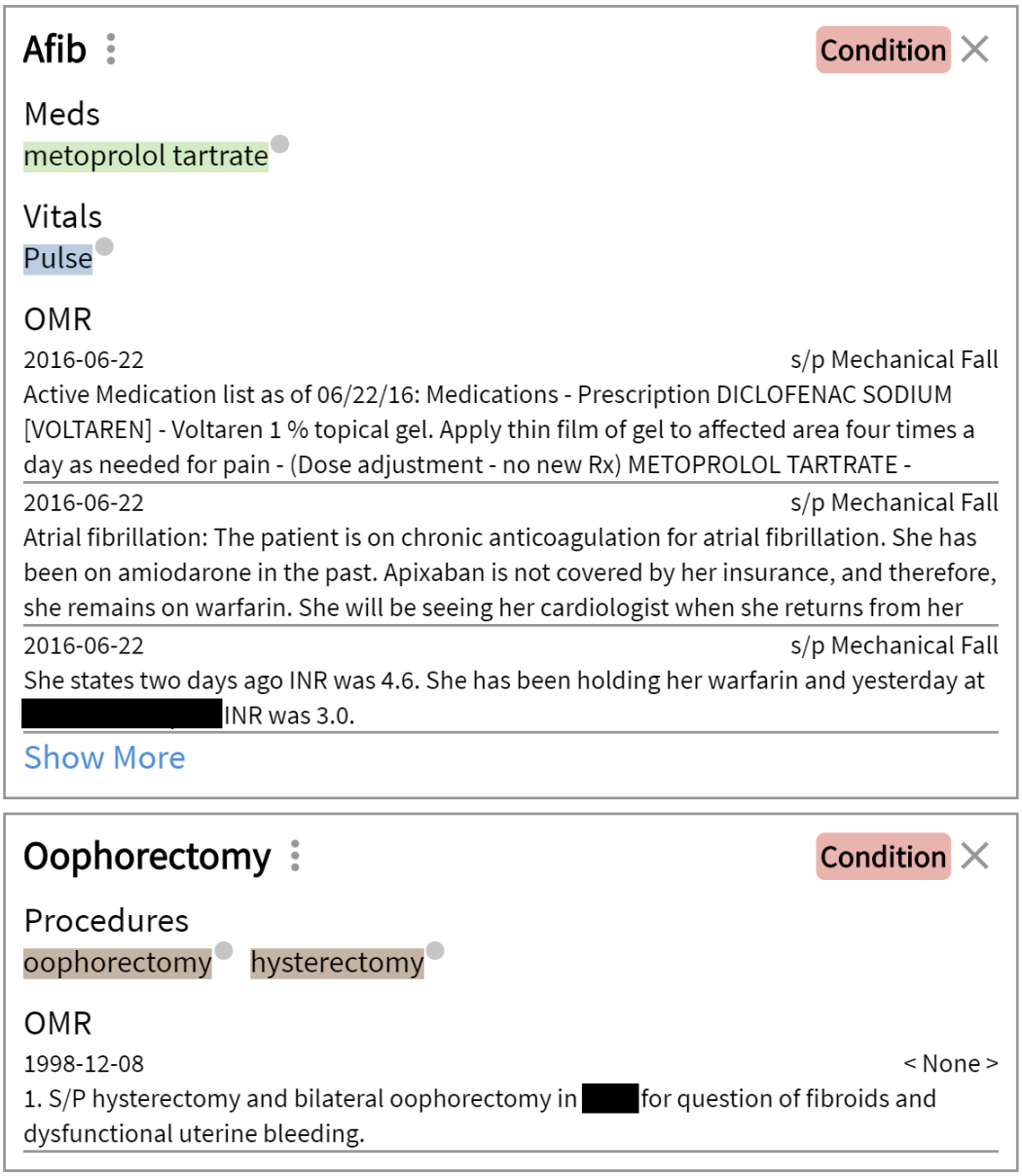


Figure 3-6: Sidebar cards house information related to given concepts. Chips within these cards can be clicked on to explore auxiliary information such as related procedures, labs, medications, vitals, and snippets of the OMR.

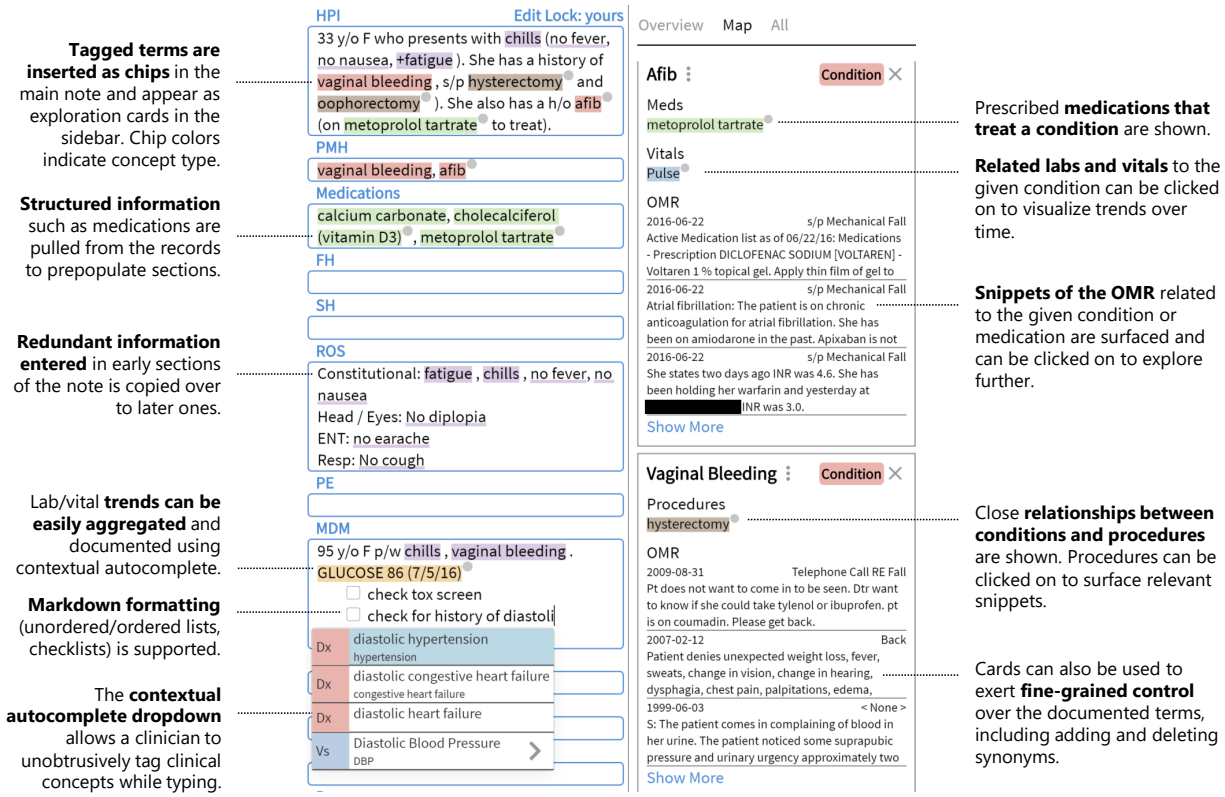


Figure 3-7: Annotated screenshot of the end-to-end EHR system, describing the data entry experience on the left and the downstream clinical decision support and information retrieval on the right.

A annotated screenshot of the complete end-to-end system is shown in Figure 3-7.

We now begin to deconstruct the tool into its constituent parts.

Chapter 4

Generating Rankings for Autocomplete

In this chapter, we discuss the learned algorithms that are the backbone of our *contextual autocomplete* functionality, which is the main data entry experience of the tool. The contextual autocomplete tool allows clinicians to easily document and annotate clinical concepts at the point-of-care – this is not only helpful to the note author, who can spend less time documenting and searching through medical records, but also to translate jargon for patients and other physicians, as well as to collect clean, labeled clinical text data for machine learning purposes.

4.1 Developing a Clinical Language Model

Ultimately, we wish to suggest a list of clinical concepts to a physician during the course of documentation. One way of doing so is to build a generative language model for clinical text that suggests chunks of text as a user types, much like Gmail's Smart Compose system [20]. In a typical generative language model, one attempts to predict the distribution $p(w_i|c_i)$ of an unknown word w_i using a *context* c_i which captures the semantic information necessary to make such a prediction. For a generative model,

c_i usually consists of a complex representation of $w_{1:i-1}$ (the words preceding w_i) and is often parameterized by a deep neural network (such a a transformer architecture). These representations are state-of-the-art for clinical language modelling [52, 124, 72]. In our framework, complex inference techniques are too slow to surface live generative suggestions with low latency in a hospital setting. In addition, predicting the general language a clinician types rather than solely clinical concepts would require note-author data to capture physician preference. As a result, we only seek to predict clinical concepts rather than the general language a clinician types.

With this constraint, we must first determine the mode by which users can browse through suggested clinical concepts and add terms to the note. Ideally, we might exploit the fact that while we are making predictions at a *concept* level, these concepts (words) are comprised of sequences of characters (letters) that a doctor must type one-by-one. We can use this character-level data to filter the suggested clinical concepts to those that match the substring the user has already typed. This paradigm lends itself well to an *autocomplete* model: We create a ranking over a fixed set of clinical concepts of C, and as a user types, we propose terms that match the written text via a dropdown. The user can then select any of the displayed suggestions. If our suggestions are particularly good, users may be able to enter terms even after typing only one or two characters of the desired concept.

There are then two components to our language model: (1) determining how to generate a ranking over C, which we refer to as *autocomplete ranking*; and (2) determining when the user is about to type a clinical concept so we can show the autocomplete dropdown, which we refer to as the *autocomplete scope*. While we must do the former without relying on the ED note text so as not to introduce system latency, scope prediction is impossible without exploiting the local structure of the note. This can be accomplished with a set of heuristics or with a sufficiently simple learned model, as we discuss in depth below. This setup is effectively creating a deconstructed, noisy language model for clinical text – by focusing on autocompleting solely clinical concepts, we can abstract away much of the messiness of physician-specific narrative text,

and by breaking down our language model in terms of ranking and scope prediction, we can exploit both historical and local contexts without introducing latency.

4.1.1 Defining Clinical Concepts

Each clinical concept $c \in C$ is defined by a list of synonyms that represent the same medical concept (e.g. `dm` and `diabetes mellitus`), one or more UMLS Concept Unique Identifiers (CUIs) that uniquely identify and normalize the concept, and finally, a concept type that gives a general sense of what the concept represents. Here, we focus on autocompleting four concept types:

- *Conditions*, which include prior diseases (e.g. `leukemia`), recurring treatments (e.g. `dialysis`), and prior procedures (e.g. `cardiac surgery`). Broadly, a condition is anything that could be used to describe a patient's medical history.
- *Symptoms*, which are indications of a patients current demeanor. Symptoms are usually observable phenomena that prompt the patient to come to the ED, such as `nausea`, `hip pain`, or `dyspnea`.
- *Medications* that a patient is on. BIDMC uses FirstDataBank's Drug Database's General Sequence Number (GSN) identifiers to store labs. GSNs can be mapped to UMLS CUIs, and drugs with the same active ingredient (e.g. coumadin and warfarin) are mapped to the same GSN and are treated as synonyms.
- *Labs* and their values, which are imperative to medical decision-making. BIDMC has its own ontology of acceptable labs, most of which have attached LOINC codes. LOINC codes can be mapped to UMLS CUIs as well.

4.1.2 Autocomplete Scope and Ranking

Our framework is a deconstructured contextual language model that decouples the questions of *when* and *how* to autocomplete. As the user is typing, we must use the

local structure of the note to determine whether a clinical concept is about to be documented. The simplest way of doing this is to a use a rule-based approach– for example, `history of [···]` indicates that a clinical concept is about to be typed next. However, notice that in the case of this toy example, we could also generate a relatively confident estimate of the concept type – `history of` is likely to be proceeded by a chronic condition, rather than a symptom, medication, or lab. Similarly, `treated with [···]` will likely be followed with a medication, and `complains of [···]` by a symptom. This leads us to a key observation: We can capitalize on the fact that we have to use local context to determine the autocomplete scope, and simultaneously determine the type of the clinical concept we want to document.

Recall that it is too expensive to update our ranking over concepts while the user is typing. In determining a ranking over concepts, all features we use as part of our context c_i must then be available before a patient and physician interact. We can use this data to generate a ranking over all clinical concept types and suggest terms to the doctor. This limits the data we can incorporate into our autocompletion models to unstructured textual data from the OMR $\mathcal{H}$, which is our glimpse into the patient's medical history; and contemporaneous clinical information T such as vitals, chief complaint, and the triage assessment.

Thus, our final framework consists of two models. We first split our set of concepts C into subsets $\{C_t\}_{t\in\Gamma}$ for each concept type t, where Γ represents the set of all concept types. C_t represents the set of concepts with type t. Our scope model $S : V^* \to \{0, 1\} \times \Gamma$ ingests an ordered list of arbitrary length of words $w \in V^*$ a clinician has typed from a vocabulary V, and makes the binary determination to display the autocomplete dropdown, as well as the clinical concept type $t \in \Gamma$ that the doctor wants to add. Then, our autocomplete ranking model $R(t, \theta, \mathcal{H})$ uses the clinical concept type t, triage information T, and historical information $\mathcal{H}$ in the OMR to create a ranking over all concepts whose type is equal to t. In practice, S and R are both learned models. Forcing R to provide a ranking over a single concept types (rather than a sole ranking over all concepts C) also makes this framework

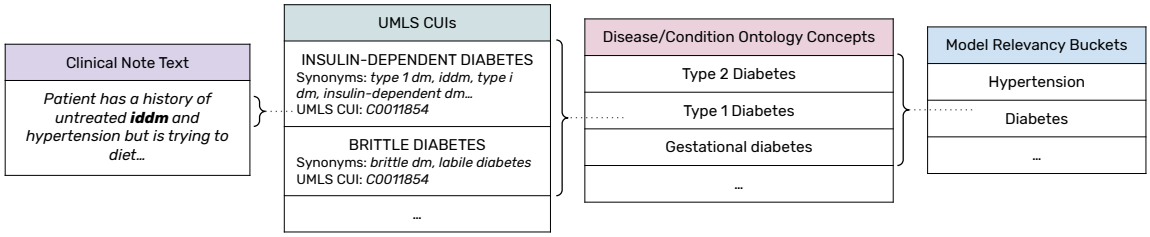


Figure 4-1: Clinical notes are normalized to UMLS, and sets of UMLS IDs (CUIs) are aggregated to create unique concepts in our ontology. Ontology entries are then grouped together in coarser model relevancy buckets as defined in Section 4.2.1.

amenable to adding other clinical concept types without having to retrain the model from scratch.

4.2 Defining the Autocomplete Problem

4.2.1 Label Generation

The goal of our contextual autocomplete is to predict terms that the doctor would type into a note given a clinical context. In order to create positive labels for this task, we must extract documented clinical concepts from medical notes through named entity recognition (NER) on the text and group them into ontologies based on concept type– we take an unsupervised NER approach to finding medical entities in the text T by normalizing it to UMLS, as we lack the labeled data for a supervised approach. An example of this is shown in Figure 4-1.

First, we restrict ourselves to a subset of the UMLS ontology and exclude terms that do not correspond to a concrete clinical concept (e.g. `Health Care Activity`). The filtered terms are then inserted into a trie data structure, which we use to identify all UMLS concepts in linear time in $|T|$. We also apply a NegEx-style negation detection algorithm as in [19] to identify and mark which of these extracted terms occurred within a negative context. After filtering out concepts that appear fewer than 50 times, we extract 8,678 remaining UMLS concepts from visit notes. We then group concepts into two categories: conditions that a patient might have a history

of, and symptoms that occur in the present medical context. We did not use our UMLS-based NER approach to generate ontologies for medications and labs, as we could source a list for both because they are structured in the BIDMC EHR system. Ambiguous acronyms such as `MS` are resolved as follows: if the term is almost always used to represent a singular concept within the ED, we default to that CUI, and otherwise ignore it. Two clinicians then independently verified these lists.

Concept disambiguation is difficult for conditions. As an example, `hyperlipidemia` and `increased LDL` are distinct UMLS concepts that encode similar semantic meanings, whose differences are not clinically meaningful in the ED and may be unknowable to the physician. To mitigate this, it was further necessary to force similar conditions to share weights by introducing a manually-curated hierarchy of UMLS terms and rolling terms up to an appropriate level of specificity such that every combined term carried the same medical meaning. A subset of this ontology for conditions is shown in Figure 4-1. The complete revised condition ontology consists of 940 entries encompassing 8,451 unique UMLS CUIs. The symptom ontology did not require this roll-up procedure and 253 entries, each representing a distinct CUI.

Condition concepts also represent varying levels of granularity, which is necessary in clinical text [116]– a doctor could use `depression`, `severe depression`, or `chronic depression` to describe a patient, but these are distinct entries in our ontology. Choosing between similar terms during documentation is currently a subjective practice that depends on the clinical scenario and user-specific preference. As such, accurately suggesting one term over another would require a knowledge of note authorship. We address this by further rolling up our ontology into a coarser set of *model relevancy buckets* which group terms corresponding to similar underlying medical concepts. We build our models to have predictive power at the level of relevance buckets, and later rank individual terms within a model relevancy bucket to suggest terms for a doctor to document. This injects a medical inductive bias that forces parameter sharing between similar concepts, thereby allowing us to leverage closely related groups of rare conditions to learn a common predictor. A subset of the 227

model relevancy buckets can be seen in Figure 4-1.

Using a trie-based extraction over a set of ontologies, we can find all unambiguous clinical concepts in a text. If a concept (whether condition, symptom, lab, or medication) appears in an ED note, it is deemed relevant for our autocomplete task and given a positive label during training of our autocomplete ranking model R. We also can use mentions of concepts and their types as training data for our scope model S. We note that while other ontologies of conditions/symptoms do exist, as in [102, 29], these concepts are not normalized to UMLS. This is key in later chapters, where we can take advantage of relationships between UMLS concepts to provide clinical decision support. In addition, this makes our framework more easily adaptable to other practices with different vocabularies.

4.2.2 Featurizing Textual Data

Our greatest sources of knowledge about the patient prior to clinician interaction lies in prior EHR notes and the triage assessment. To featurize prior EHR documents, we run the NER and hierarchical roll-up algorithms from Section 4.2.1. The result of this is a mapping from a clinical text T to a set of UMLS-mapped clinical concepts mentioned in the text, as well as a coarser representation of the types of conditions incorporated into the note via model relevancy buckets. To encode triage assessments, we simply use a standard term frequency-inverse document frequency (TF-IDF) encoder to capture a normalized bag-of-words representation of the text. Although we experimented with other ways of representing textual data e.g. via BERT-based contextual embeddings [30] or word2vec representations of concepts [125], finding high-quality embeddings for infrequent medical concepts that appeared in our ontologies was difficult.

4.2.3 Alternate NER Approaches

In order to confirm that our UMLS-mapped trie-based extraction of clinical concepts was reasonably accurate and performant, we also consider a few alternate ways of perform clinical NER on ED note text. We restrict our search to techniques that normalize to UMLS, as this is a key benefit of our system that makes it extendable.

First, we attempted to extract concepts directly from the raw text, without normalizing to an ontology. We did this by extracting common unigrams and bigrams and removing common stopwords (`and`, `to`). We manually went through the 1,000 most common terms to confirm they were reflected in our UMLS-mapped ontology of conditions, and added a handful of terms that were missing: `hld` as a synonym for hyperlipidemia, `hep c` as a synonym for hepatitis C, `pna` for pneumonia, etc. We note that ontologies are always a work in progress and that our current system provides doctors with the ability to submit ontology modifications that can then be reviewed.

We compare our trie-based extraction against three baselines:

- `cTakes`, or the Mayo clinical Text Analysis and Knowledge Extraction System, which combines rule-based and simple machine learning techniques to extract and normalize concepts to UMLS [105]. cTakes is an older system that often misses clinical abbreviations [99]. We limit the cTakes vocabulary to UMLS concepts in our ontology to provide a fair comparison.

- `scispaCy`, which is a Python biomedical text processing library built on top of `spaCy` [84]. It contains neural entity extraction trained on biomedical corpora using a bidirectional-LSTM with a conditional random field (CRF) layer as proposed in Lample et. al [64]. `scispaCy` identifies clinical and biomedical terms on the text first with the entity recognition model, and then retroactively maps this to UMLS using a string match over synonyms.

- BERT-based clinical entity extraction models such as [5], which combine a trans-

System	Latency (seconds)	Comments
Trie-based	0.8	Ours, poor disambiguation for the few overloaded concepts
cTakes	37	Provides virtually the same extraction as the trie-based procedure, but with certainty/polarity scores
scispaCy	19.5	Bulk of the time spent on mapping extracted terms to UMLS. Some acronyms were not disambiguated, e.g. `dm` was extracted as both `diabetes mellitus` and `double miutes`
DistilBERT	489	No extraction, just passing windowed snippets of the text through a compact transformer

Table 4.1: Comparing NER approaches on OMR notes both by latency and by qualitative ability to extract concepts well. Latency is measured by time to process 100 randomly chosen OMR notes.

former architecture with CRFs and other layers that are good at entity identification. These models are considered state-of-the-art in neural entity extraction, but are fairly slow and cannot easily run on our servers, which we discuss below. While we cannot easily compare to [5] due to the lack of labeled data to train the deep model, we measure latency of running BERT on a sequence of clinical notes as a proof of concept. We use DistilBERT as our base BERT model because of its compactness [104], and train on a custom vocabulary which is smaller than that of the original BERT model [30].

While it is difficult to quantitatively compare these methods because we lack gold-standard entity labels for our dataset, we find that the trie-based method is significantly faster than our three other comparisons with little to no loss in recognition quality.

Note that all of the learned models also preclude us from making easy changes to our ontology – it is difficult to retrain these models without sufficient labeled data of a given clinical concept, which may not exist. On the other hand, our trie-based approach is reasonably fast and trivial to extend. We find that it is suitable for our purposes.

4.3 Autocompletion Model by Concept Type

We frame contextual autocomplete as a hierarchical, human-in-the-loop language model that suggests clinical concepts to document as a physician is typing. We leverage four pieces of data to form our context $c_i = [w_{1:i-1}, T, \mathcal{H}, V]$; namely, the text so far, the triage assessment, past OMR notes, and the patient's triage vitals. Our scope model not only uses $w_{1:i-1}$ to determine when to show our autocomplete but also to determine the concept type of the incoming clinical concept – whether it is a condition, symptom, medication, or lab. This is then then used in our autocomplete ranking model R.

Calling an inference step of our model R each time a word is written or removed is prohibitive in terms of latency, so through this technique, we can exploit the local structure of $w_{1:i-1}$ to reduce the number of clinical concepts we need to predict over, while learning how to use the nuanced information in $T, \mathcal{H}, V$ to inform what concepts might be important. Inference on R thus only needs to be run once per patient, for each concept type. This can happen as soon as triage information is entered for the patient, so that the rankings are ready when a doctor starts to type his/her clinical note. We generate four term-wise rankings for each concept type, and stack the suggested rankings for each of the concept types to generate a total ranking. In practice, we filter these rankings to entries with any synonyms that match the typed query a doctor has entered. The doctor can either continue to type or select a term, which is then inserted into the note as a tagged concept using the synonym that they intended – as an example, typing `ht` might give `hypertension` as a suggestion because of its synonym `htn`, and if a doctor chooses to autocomplete, we insert `htn` to preserve intended note vocabulary.

We outline our concept-specific ranking models before delving into the details of each:

1. Conditions: We learn a mapping from the triage text and the clinical concepts mined from the EHR to a ranked list of relevant prior conditions that the doctor might want to document. This autocompletion model is primarily used to write

the History of Present Illness (HPI) sections of notes, where physicians document past medical history that is relevant to the current patient presentation. We find that vitals have little to no predictive power in this model.

2. Symptoms: We learn a mapping from the triage text, chief complaint, and vitals to a ranked list of relevant symptoms that the patient currently presents with. We do not include information from the patient's past medical record in our predictions because a patient's current presentation is only loosely related to prior visits.

3. Labs: We simply list labs by their recorded frequency in $\mathcal{H}$, rather than learning a mapping. The space of labs is much smaller than the space of symptoms or conditions, so we find that a frequency-based ranking is nearly optimal in practice.

4. Medications: As with labs, we rank by frequency for the same reasons.

4.3.1 Autocompleting Conditions

Documenting relevant patient history is often an arduous task for physicians in the ED. Doctors typically read a patient's triage assessment and then search through a patient's OMR on an ad-hoc basis to try and contextualize the current visit with the patient's background. In our dataset, there is a median of 65 distinct conditions mentioned in a patient's EHR, but on average, only five of these concepts are then documented in the ED clinical note. In addition, around a quarter of the patients in our dataset do not have any prior records on file; in these cases, doctors can guess relevant conditions to inquire about based on the triage text and chief complaint alone.

This leads to key model desiderata: first, we must be able to recover an intelligent ranking over concepts even in the absence of prior medical notes using triage information alone. Second, we seek to learn a single multilabel ranking over all possible model relevancy buckets in order to produce a globally calibrated model. Our model

first learns a ranking over the coarse model relevancy buckets, and then recovers a ranking over individual condition concepts to mention in the note.

We use a shallow, dual-branch neural network architecture to combine a context c_i consisting of a TF-IDF representation of the triage text T and a feature vector indicating the binary presence $\mathbf{1}[b \in \mathcal{H}]$ of each model relevancy bucket b in prior OMR notes $\mathcal{H}$. In more detail, the neural network takes in two inputs:

1. A Term Frequency-Inverse Document Frequency (TF-IDF) representation of the triage text using unigrams and bigrams. Vocabulary size is close to 22,000.

2. The binary presence of different *model relevancy buckets* (as defined in Section 4.2.1) in the patient's prior medical history. This is a length-227 binary vector.

These two inputs are both passed through two separate dense layers with ReLU activation, concatenated and passed through another dense layer, and then finally passed through element-wise sigmoid activations to generate probabilities per class. We train this model with stochastic gradient descent using a cross entropy loss function.

We recover a term-wise ranking by sorting each term First by whether it appears in the OMR, then by the rank of its relevance bucket, and finally by its empirical frequency of occurring in the data to resolve ties. In this way, we create a single architecture that predicts $P(b|T, \mathcal{H})$, or the probability of b being relevant given the triage information and prior history, for all b simultaneously and thereby suggest conditions to document for patients both with and without a prior medical history.

We also compare against three baselines:

1. *One vs. Rest Logistic Regression on Triage Text*: We build a model based solely on T. For each model relevancy bucket b, we estimate the $P(b|T)$ via a logistic regression model trained on a TF-IDF representation of T to predict if any term in b was mentioned in the corresponding clinical note. We randomly select notes without any mention of b to generate negative samples. To make a prediction for a given patient, we then rank relevance buckets b by $P(b|T)$. To recover

a term-wise ranking, we sort each term first by the rank of its corresponding relevance bucket and by its empirical frequency in clinical notes.

2. *One vs. Rest Logistic Regression on Triage Text, EHR*: As above, we train a logistic regression model on T for each model relevancy bucket b. However, when predicting $P(b|T)$, we restrict ourselves to train on samples where b is mentioned in $\mathcal{H}$. That is, our model predicts the probability $P(b|T, 1[b \in \mathcal{H}])$. We assume $P(b|T, 0) = \epsilon_b$ for a small but nonzero ϵ_b, or that if if a term does not appear in a patient's EHR, it is unlikely that it will be documented in the present note. To recover $P(b|T)$, we multiply by an empirically computed prior probability $P(1[b \in \mathcal{H}])$ of each bucket being mentioned in the EHR. We recover a term-wise ranking using the same key as the previous method. The leak probability ϵ_b allows us to rank buckets that are not present in the EHR by their empirical probabilities alone, giving us predictive power for patients without any prior history.

3. *Augmented One vs. Rest Logistic Regression on Triage Text, EHR*: We experiment with feature-engineering approaches to include signals from the EHR in our model covariates. In particular, we augment the feature space with a representation D of how many days it has been since b was mentioned in the EHR, and compute $P(b|T, D, 1[b \in \mathcal{H}])$ via logistic regression. In order to force this input variable to conform to a normal distribution, we transform the delay times by assuming mentions follow a Poisson process and concluding that delay times should be exponentially distributed. We follow the same empirical reweighting and term-wise ranking procedure as in the previous model.

4.3.2 Autocompleting Symptoms

Based on discussions with clinicians as well as qualitative analyses within our slice of ED data, we find that the symptoms that a doctor asks a patient about and subsequently records are primarily rule-based. A chief complaint of dyspnea at triage-

time, for example, might prompt the doctor to inquire about dyspnea (reaffirming that it is still a concern), chest pain, coughing, etc. Consequently, the models we develop for symptom autocompletion use only the chief complaint and triage vitals as covariates. We perform ablation tests with all of our models to confirm that adding in a bag-of-words representation of the triage text did not increase performance, and develop four schemes to map chief complaints and vitals to a ranking over symptoms:

1. *Empirical Conditioning on Chief Complaint:* For a given chief complaint c, we empirically calculate $P(s|c)$ for each s in the set of symptoms S, and rank each symptom by this probability.

2. *Empirical Conditioning on Chief Complaint, Vital:* For a given chief complaint c and a list of vitals V, we calculate the single vital $v \in V$ that is most abnormal. Abnormality is defined as the percentile deviation from the population median of the vital value. We then encode v as a categorical variable $b(v)$ based on medical guidelines about the given vital (for example, heart rate vitals are placed into one of three buckets: `LOW HR`, `NORMAL HR`, and `HIGH HR`). Full details about the bucketization procedure are given below. Finally, we empirically calculate $P(s|c, b(v))$ for each $s \in S$, and rank each symptom by this probability.

3. *One vs. Rest Logistic Regression:* For each symptom $s \in S$, we train a logistic regression model mapping the chief complaint and vital values to whether s appears in the ED note corresponding to that visit. Then, we rank the output probabilities for each symptom.

4. *One vs. Rest Naive Bayes:* For each symptom $s \in S$, we train a Naive Bayes classifier mapping the chief complaint and vital values to whether s appears in the ED note corresponding to that visit. Then, we rank the output probabilities for each symptom.

In practice, we find that the second scheme performs best and we use this for deployment. Comparative performance for these models is detailed in Section 4.4.

Bucketization of Vitals

As described above in Section 4.3.2, our best model for predicting a ranked list of relevant symptoms to document relied on a categorical featurization of triage vitals. The model simply uses the empiric frequencies of symptoms documented in a note, conditioned on the chief complaint c and a categorical representation $b(v)$ of the most abnormal vital v. We used medical guidelines to determine cutoffs for each vital as follows:

- *Temperature:* Temperatures above 100.4° are considered `HIGH` as they are medical-grade fevers. Temperatures below 97° are considered `LOW` as they are hypothermic. Otherwise, a temperature is considered `NORMAL`.
- *Respiratory rate:* A respiratory rate above 20 breaths per minute is considered `HIGH`, as per [28]. A respiratory rate below 12 breaths per minute is considered `LOW`. Otherwise, the respiratory rate is considered `NORMAL`.
- *Blood oxygen level*: A pulse oximeter reading below 95% is considered `LOW` as per Mayo Clinic guidelines. Otherwise, the reading is considered `NORMAL`.
- *Heart rate:* A heart rate above 100 beats per minute (bpm) is considered `TACHYCARDIC`. A heart rate below 60 is considered `BRADYCARDIC`. Otherwise, it is considered `NORMAL`.
- *Blood pressure:* Based on guidelines set by the American Heart Association, a systolic BP under 120 mmHg and a diastolic BP under 80 mmHg constitutes a `NORMAL` BP. If the diastolic BP is under 80 mmHg but the systolic BP is between 120-130 mmHg, it is considered `ELEVATED` blood pressure. If the systolic BP is under 140 mmHg and the diastolic blood pressure is under 90 mmHg, this is characterized as `STAGE 1 HYPERTENSION`. Otherwise, if either reading is higher, it is `STAGE 2 HYPERTENSION`.
- *Age:* Based on the age distribution of patients in the hospital, we bucketized patients into six groups: `CHILD` (i.e. below 18), `18-33`, `34-48`, `48-64`, `64-77`,

and `78+`.

4.3.3 Autocompleting Labs and Medications

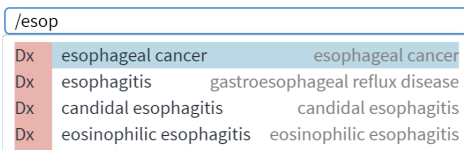
Autocompleting labs and medications is different from symptoms and conditions in a few marked ways. A patient's medical record contains structured information about prior lab tests and values, as well as medications and their dosages prescribed in the past. This is in contrast to symptoms and conditions which are almost always referenced in unstructured notes or free text. Concept disambiguation is less pertinent because there are structured representations of labs and medications, and there are already semi-structured lists of labs and medications that exist in clinical records. The primary value-add for physicians to tag a mention of a lab/medication in a note is instead to *enable immediate information retrieval*. Tagging `HCT`, for example, can prompt the visualization or insertion of a patient's hematocrit trend. We thus add lab and medication autocompletion to be thorough in our data collection, and use a frequency-based autocompletion for both data types.

A next step would be to use the structured data collected with our tool to build a learned model to predict over labs and medications, rather than using a frequency-based approach.

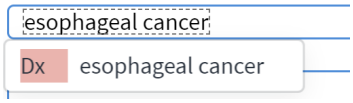
4.4 Contextual Autocomplete Results

A physician uses contextual autocomplete by naturally typing a note and either automatically or retroactively completing clinical phrases that are then rendered as tagged concepts. We describe the user experience of the tool with a screenshot in Figure 3-2, and examine how it reduces clinical documentation burden in practice.

In order to evaluate our ranking model R independently, we set our scope detection model S to be rule-based at first. The heuristics we used are detailed in the next chapter and are compared to the final learned model for S. At a high level, the



(a) Manual autocompletion trigger



(b) Retroactive tagging

Figure 4-2: Screenshots of our backup data capture strategies in the case that the autocompletion scope detection fails. (a) Users can insert a slash character (`/`), which acts as a manual trigger to force autocompletion. (b) Users can retroactively accept tags for candidate concepts that they typed but did not autocomplete.

rule-based approach simply looks for a list of pre-defined phrases that act as scope triggers (such as `complains of`). Each trigger is mapped to the concept type that is likely to follow it. We also support two fallback data capture methods for when our algorithm fails. First, a user can start an autocomplete scope with a manual trigger. In addition, if the user does not type the manual trigger, we use an Aho-Corasick keyword detection algorithm to efficiently map exact string matches in the text with clinical concepts to our ontology [3]. Any matches are displayed as potential tags which doctors can manually confirm if desired. A screenshot depicting these backup data capture strategies can be seen in Figure 4-2. We analyze how often these mechanisms are exercised in practice below.

4.4.1 Performance and Usability

From an information retrieval perspective, we can analyze the quality of our ranked list of suggested clinical concepts by using two standard metrics: the *mean reciprocal rank* (MRR) and *mean average precision* (MAP). Consider an ordered ranking $\mathcal{R} = \{r_1, r_2, \cdots\}$ of suggested terms and a ground truth set of terms that the clinician wants to document denoted by $T = \{r_{\pi(1)}, r_{\pi(2)}, \cdots\}$. We define the MRR of these suggestions as

$$MRR = \frac{1}{|T|} \sum_{\{r_i \in \mathcal{R} | r_i \in T\}} (\max(1, i - |T|))^{-1}$$

In other words, this measures the average excess rank of the suggested terms that actually occur in the ground-truth terms the clinician wants to document. An MRR

of 1 indicates that k desired terms were in the top k suggestions. The MAP score, in contrast, measures the average proportion of ground-truth terms that occur in the top k suggested terms as k varies:

$$MAP = \frac{1}{|T|} \sum_{k=1}^{|T|} AveP(k)$$

where $AveP(k)$ represents average precision of the top k suggested terms. A MAP of 1 indicates perfect precision.

Because the primary goal of this tool is to improve documentation efficiency, we also define the *keystroke burden* as the number of keystrokes the clinician needs to type until he/she autocompletes and inserts a desired term. This usability metric inherently encompasses the quality of our information retrieval in its calculation while also incorporating real-world behavior – there may be a delay between a term being suggested first and when a clinician actually autocompletes the term.

We compare the MRR, MAP, and keystroke burden of our contextual autocompletion tool rankings against two naive baselines: spell-based autocompletion (ranking terms alphabetically) and frequency-based autocompletion (ranking terms by frequency).

Retrospective Evaluation on Clinical Notes

Before deploying our autocompletion models in a live setting, we evaluated the quality of our suggested rankings via retrospective annotation of the clinical notes we had on file. In particular, we measured performance broken down by concept type, as well as the efficacy of our autocompletion scope and type detection algorithms.

To generate our evaluation set, we extract medical concepts from 25,000 clinical notes by the technique outlined in Section 4.2.1. Using the order in which concepts were suggested, we first measure MRR, MAP, and keystroke burden assuming perfect scope and type detection, broken down by the four concept types. Results are shown in Table 4.2. We see the largest gain in using a contextual model for conditions, because

the space of terms is large and the richness of the OMR greatly influences documentation. Within the contextual models for predicting prior conditions, the dual-branched neural network outperforms others primarily because it is predictive even for patients with no medical history on file. On the other hand, when documenting symptoms, a model that ranks symptoms by their empirical frequency (conditioning on the chief complaint and the most abnormal vital) performs best.

To quantify the ease of documentation using our rule-based autocomplete scope algorithm, we also measure MRR, MAP, and keystroke burden when typing HPI sections of notes. On average, there are 6.8 documented clinical concepts per HPI section. As HPI sections contain conditions, symptoms, and occasionally medications, we evaluate our autocomplete type predictions (whether the concept type was guessed correctly) on a realistic range of concept types. Of the extracted clinical concepts in HPI sections, 46% of terms were autocompleted automatically without a manual trigger, and in 77% of those cases, we guessed concept type correctly as well. As a result, the MRR of automatically-detected autocompleted terms is 0.35. Even in cases where the doctor is forced to insert a manual trigger to autocomplete a term, we still greatly decrease the documentation burden on doctors as shown in Tables 4.2 and 4.3. These manually prompted scenarios can be mitigated as a doctor learns and adapts to the triggers of the system, which we elaborate on later.

Model Type	MRR ↑
Conditions	
One vs. Rest Logistic Regression on T	0.09 ± 0.02
OvR LR on T, EHR	0.15 ± 0.02
Augmented OvR LR on T, EHR	0.17 ± 0.01
Dual-branched neural network	**0.28** ± 0.01
Symptoms	
Empirical Conditioning on Chief Complaint	0.39 ± 0.01
Empirical Conditioning on Chief Complaint, Vital	**0.42** ± 0.01
One vs. Rest Logistic Regression	0.16 ± 0.01
One vs. Rest Naive Bayes	0.27 ± 0.02

(a) Comparison of MRR between contextual autocompletion models

Model Type	Autocomplete Type		
	Spell	Frequency	Contextual
Conditions	0.01 ± 0.001	0.08 ± 0.01	**0.28** ± 0.01
Symptoms	0.05 ± 0.001	0.27 ± 0.01	**0.42** ± 0.01
Labs	0.01 ± 0.001	0.40 ± 0.01	N/A
Medications	0.02 ± 0.001	0.02 ± 0.001	N/A
Overall	0.01 ± 0.001	0.19 ± 0.03	**0.29** ± 0.05

(b) Comparison of MRR across autocomplete types

Table 4.2: Retrospective Evaluation of MRR using Contextual Autocomplete. We report average MRR ($\pm$95% confidence interval of the mean) for each of our learned contextual autocomplete models, and compare our best models (dual-branched neural network for conditions, empirical conditioning on the chief complaint and most abnormal vital for symptoms) to spell-based and frequency-based baselines, both for specific concept types as well as overall using our scope and type prediction algorithms. Calculated across 25,000 visits.

Model Type	Keystroke Burden ↓	MAP ↑
Conditions		
Frequency-based baseline	3.44 ± 0.09	0.08 ± 0.01
One vs. Rest Logistic Regression on triage text T	3.02 ± 0.09	0.08 ± 0.02
OvR LR on T, EHR	2.81 ± 0.08	0.15 ± 0.02
Augmented OvR LR on T, EHR	2.71 ± 0.08	0.16 ± 0.01
Dual-branched neural network	**2.57** ± 0.07	**0.27** ± 0.02
Symptoms		
Empirical Conditioning on Chief Complaint	2.19± 0.04	0.41 ± 0.01
Empirical Conditioning on Chief Complaint, Vital	**2.09** ± 0.03	**0.44** ± 0.01
One vs. Rest Logistic Regression	2.74± 0.02	0.16 ± 0.01
One vs. Rest Naive Bayes	2.51 ± 0.03	0.30 ± 0.01
Labs (ranked by frequency)	0.092 ± 0.03	0.39 ± 0.01
Medications (ranked by frequency)	3.28 ± 0.04	0.03 ± 0.01
Overall with autocomplete scope/type detection	3.13 ± 0.05	0.27 ± 0.06

Table 4.3: Retrospective Evaluation of Keystroke Burden and MAP using Contextual Autocompletion. We report the mean keystroke burden/MAP for the contextual autocomplete models we prototyped for each concept type, following the conventions of Figure 4.2.

Documentation in the Wild: Live Evaluation

We compare keystroke burden between a contextual model and no autocomplete in Table 4.4. In our live evaluation, a single physician wrote 40 notes using our system over two shifts. In practice, an average of 8.38 terms are tagged per note, and we reduce overall keystroke burden for these clinical concepts by approximately 67%, with clear gains in using our model irrespective of note section or concept type. 53% of the tagged clinical concepts were autocompleted without a retroactive label. Within these concepts, the autocomplete scope algorithm was correct 77% of the time, indicating that we were able to guess the concept type correctly.

4.4.2 Sensitivity Analysis

Concept frequency influences the efficacy of our contextual autocomplete model of conditions. The biggest wins in the model occur with the group of conditions in the middle of the frequency distribution – `renal insufficiency`, for example, is an infrequent but not rare term that will almost certainly be documented in a note if it appears in the patient's history. The symptom contextual autocompletion model, on the other hand, is generally agnostic to concept frequency because the space of symptoms is much smaller and the distribution of symptoms is less skewed than that of conditions.

In addition, the presence of prior medical history has significant impact on contextual autocompletion performance for conditions– as shown in Table 4.5, we see greater reduction in documentation burden if the patient has prior EHR. However, our contextual model and a frequency-based autocompletion model perform similarly for concepts that are not mentioned in the EHR despite the person having some prior medical history– this can largely be attributed to the inherent bias of our ranking scheme, which preferentially orders terms mentioned in the EHR above those that are not.

Subset	Autocompletion Type	
	None	Contextual
Overall	11.85 ±1.94	4.32 ±0.43
By Note Section		
History of Present Illness	12.36 ±2.16	4.57 ±0.87
Past Medical History	11.41 ±2.09	2.94 ±0.68
Medical Decision Making	10.27 ±3.18	4.08 ±0.49
By Concept Type		
Conditions	13.08 ±1.72	4.34 ±1.49
Symptoms	8.5 ±2.18	4.53 ±1.00
Labs	10.33 ±5.76	2.06 ±0.88
Medications	9.27 ±1.97	4.27 ±1.34

Table 4.4: Live Evaluation of Contextual Autocomplete Models. Mean keystroke burden for autocompleted concepts (± 95% CI from mean), measured across 40 notes written live by a single physician over two shifts. Performance is also broken down by note section, as well as concept type.

Mean Keystrokes Saved per Condition Concept			
	Uncommon Concepts	Median Concepts	Common Concepts
With no past EHR at hospital	0.63 ±0.42	0.81 ±0.50	0.47 ±0.20
With prior mention of concept in EHR	2.64 ±0.65	2.02 ±0.38	1.40 ±0.16

Table 4.5: Number of keystrokes saved by our contextual model compared to a frequency-based baseline (±95% CI of the mean) for conditions. Performance was stratified by concept frequency (by terciles) and by available medical history.

Concept	Most Predictive Triage Tokens	Most Predictive Model Relevancy Buckets
Dementia	dementia, abrasions, fell, home, fall, neuro, son, ...	dementia, neurodegenerative diseases
Bronchitis	pna, pneumonia, cough, sob, hemoptysis, sputum, ...	pneumonia, chronic lung disease
Prostate cancer	ca, mass, chemo, lymphoma, melanoma, cll, tumor, ..	cancers, prostatectomy
CHF	chf, chest, sob, cp, cough, syncope, fall, ...	heart failure, heart attacks, hypertension, afib
Diabetes	bs, fsbs, glucose, iddm, sugars, toe, finger, ...	diabetes, hyperlipidemia, diabetic neuropathies, gastroparesis

Table 4.6: Predictive features for selected condition concepts, using a linear approximation to our contextual model for conditions. Inputs to the model are a TF-IDF representation of the triage text as well as the presence of coarse-grained model relevancy buckets in a patient's prior medical record, as defined in Section 4.2.1.

4.4.3 Interpreting Autocompletion of Prior Conditions

Because our contextual model for conditions learns a ranking from a representation of the triage text and medical history, it is naturally more sensitive to changes in input than our contextual model for symptoms. Here, we dig further into what drives model predictions.

Performance By Concept

Our multi-label model predicts the binary relevance of each model relevancy bucket. To better interpret relevancy predictions on a per-bucket level, we approximate our model for a specific relevancy bucket b with a linear function of the inputs. This is done by fitting a L_1-regularized linear approximation between the features and the logits generated by the model for bucket b to surface highly-weighted features [49]. In Table 4.6, we provide examples of the top-weighted positive features in the linear approximations to models for five selected concepts. Overall ranking performance by MRR for these concepts is in Figure 4-3. Interestingly, while all of the chosen concepts

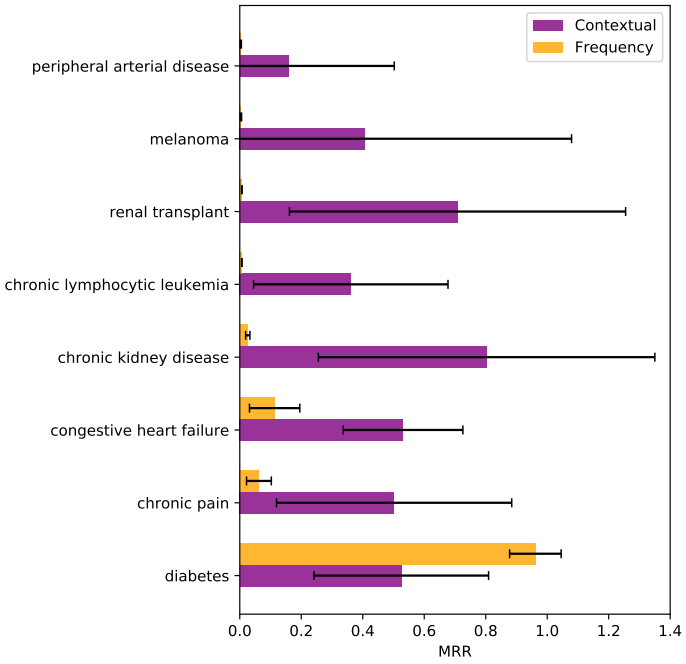


Figure 4-3: Mean MRR for five conditions (± 95% CI from mean) using contextual and frequency-based autocompletion. Concepts were chosen to get representative samples of the data.

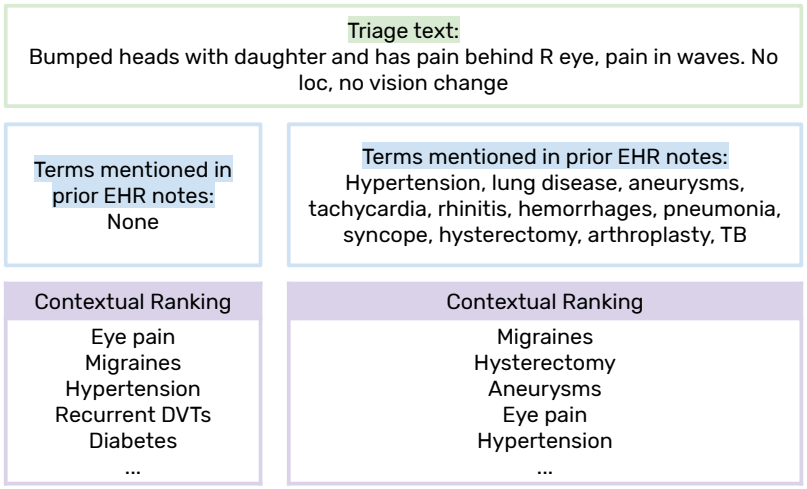
relied on medically meaningful tokens present in the triage text, the linear models for diabetes and congestive heart failure both used the presence of many model relevancy buckets, whereas the other three concepts only relied on a few. This is likely because the model always relies on triage text but can give predictions even in the absence of prior medical history, and as the linear approximation to our model encourages sparsity, only highly predictive model relevancy buckets will be chosen as features. A frequency-based baseline outperforms our learned model only for extremely common conditions like hypertension and diabetes.

Qualitative Evaluation & Readability

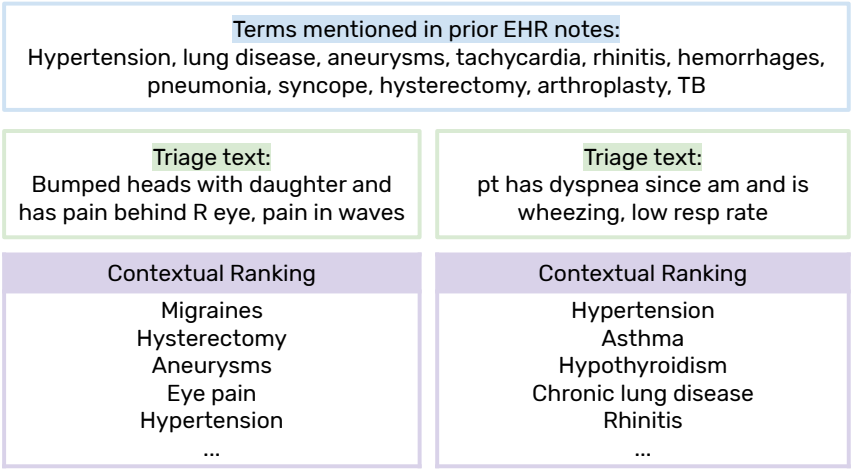
We qualitatively evaluate rankings over conditions to better understand model decisions. As can be seen in the selected examples in Figure 4-4, both the presence of OMR notes as well as specific types of words mentioned in the triage note can have great impact on the rankings, which are much more context-specific than frequency-based rankings. Chronic conditions mentioned in a patient's medical history are highly ranked even if they are not directly related to the present medical context, because they are likely to be documented regardless. For example, in Figure 4-4a, two patients have identical triage text but different medical histories– consequently, `hysterectomy` is highly ranked for one. Of course, the triage note still governs the overall theme of the most highly ranked terms; in Figure 4-4b, two patients with identical medical histories but differing chief complaints have vastly different context-specific rankings.

4.5 Summary

The contextual autocomplete ranking models outlined here describe a way to automatically tag clinical concepts in text while simultaneously reducing the documentation burden on physicians. The ablation tests we carried out show that using a few features (primarily representations of medical histories) can result in performant pre-



(a) Effect of patient history on contextual rankings.



(b) Effect of triage note on contextual rankings.

Figure 4-4: Case Studies of Autocomplete Rankings over Conditions

dictive models for documentation. This is a critical advantage of our system because EHR data is often very sparse– patients can enter the ED with no prior medical history, yet we can still glean information from the triage assessment to represent a patient state. Our strategies also obviate the need for complex data imputation schemes.

With this set-up, we use our ranking model R to generate lists of relevant concepts that are stratified by concept type. We evaluated the quality of R using MRR, MAP, and a keystroke burden usability metric. However, measuring the true benefit our tool provides also requires evaluating our scope detection model S, which predicts when to display the autocomplete dropdown and if so, which concept-type ranking. In this chapter, we evaluated R while setting S to use the simplest rule-based heuristic possible. In the next chapter, we compare this to a series of learned models.

4.5.1 Future Direction: Dynamic Autocomplete Rankings

A next iteration of contextual autocomplete should dynamically update suggested terms to document using already-tagged terms in the note. Tagging `afib`, for example, might indicate that there is a high likelihood of the doctor typing an anticoagulant next. Using live data collected from the deployed tool, we can use early drafts of a clinical note to influence the medical context for later autocompletion suggestions. We can also clarify patterns of redundant data entry by examining where the same underlying medical concept is repeated in the note, with the eventual goal of learning and auto-inserting necessary repetitious documentation. These dynamic updates introduce a significant latency on the client-side UI to perform online inference as words are typed, so this may not be feasible for all systems and thus we did not consider it for this thesis.

Chapter 5

Triggering Autocomplete Scope

The second part of developing our deconstructed language model for clinical concepts it to determine when to show the autocomplete display, or its *scope*. In Chapter 4, we remarked that given that we had to use the local structure of the note to determine autocomplete scope, we could also use the same context to make a prediction as to the autocomplete *type*– whether to show the ranked list of conditions, symptoms, medications, or labs. We denote our autocomplete scope model by S, which makes a binary prediction as to whether to display autocomplete, as well as the likelihood of it being each concept type. S must be run every time a user updates a live EHR document to predict in real-time if the autocomplete display should be shown at the user's latest cursor position.

It isn't obvious why S is necessary at all, in that from a data collection perspective, the best way is to always show the autocomplete dropdown and thus never miss a tagged concept. However, based on preliminary user studies, doctors find the flashiness of the autocomplete annoying when they are not documenting clinical concepts. Moreover, we must use the local context to determine the autocomplete concept type regardless.

The biggest constraint on S is its inference time. Because we make calls to S every time a user appends a word to the text, the model latency has to be faster than the user's typing rate. As an ideal upperbound, response times of of 16-17ms feel

smoothest, which matches the refresh rate of most screens (about 60 frames per second) [85]. Note that this is the acceptable end-to-end latency for a system, which includes text preprocessing, model inference, parsing model output, and rendering.

We now discuss two ways of parameterizing S– via fixed rules, as well as through a learned model.

5.1 Rule-Based Triggers

Using manual triggers for autocompletion can establish consistent system behavior for physicians, and create notes that are concise. We found that clinicians tended to quickly adapt to common learn triggers such as `complains of` and `history of`. As an example, one clinical note in our dataset began with the phrase `patient has a history of abdominal pain which seems recurrent`, whereas our system would autocomplete to `patient has a history of` *`chronic abdominal pain`*. Of course, rule-based triggers are never all-encompassing and pose a learning curve to physicians, which is why we ultimately opt for a learned model.

In our rule-based approach, we first define a default concept-type ranking per note section. For example, in HPI, the majority of documented content pertains to historical conditions and some current symptoms/medications, so the default ordering is `CONDITION, SYMPTOM, MEDICATION, LAB`. In contrast, in a Physical Exam section, clinicians document symptoms more than chronic conditions, so the default ordering is `SYMPTOM, CONDITION, MEDICATION, LAB`. We then establish certain key phrases to act as autocomplete triggers if they are likely followed by a clinical concept. We curate a list of common trigger phrases (e.g. `presents with`, `history of`) and map them to the concept type that follows them– `presents with` is mapped to `SYMPTOM`, and `history of` to `CONDITION`. Using these, we create a NegEx-inspired algorithm to predict both autocomplete type and scope [19]. The algorithm greedily uses keywords that act as autocomplete triggers, and is run and updated as a physician types a clinical note. First, we initialize the scope and

type of our autocomplete to be null. Then, for each word w in the text, we update the scope accordingly:

- If w is part of a autocomplete trigger phrase such as `presents with`, we turn the autocomplete scope on and suggest terms to the user. We set the autocomplete type based on the trigger (`presents with` maps to `SYMPTOM`.)
- If w is a continuation token such as `and`, `or`, or `,`, we maintain the current scope and autocompletion type.
- If w is part of a tagged concept c, we turn the autocompletion scope on, and set the autocompletion type to the concept type of c.
- Otherwise, w is treated as a stopword, in which case the autocompletion scope is turned off.

With this framework, the autocompletion scope and type is greedily set using a simple parsing algorithm that is rerun as the user types a new word. The rule-based approach establishes consistency and simplicity for the doctor– behavior is easy to predict, and the autocomplete dropdown is not constantly displayed, which can be a nuisance to doctors.

5.2 Learned Triggers: The Setup

S can also be a learned model, which can enable us to exploit local context in a more intelligent way. For example, a user's narration style can affect the autocomplete scope, and learned models can capture these longer-term dependencies. In addition, while the false positive rate of the rule-based algorithm was low, the false negative rate is high. False negatives are more costly in our framework, as they result in a missed opportunity to capture structured data. Essentially, we seek to maximize precision subject to high recall.

A learned model of S would output a soft probabilistic estimate p of whether or not to display the autocomplete dropdown. Determining when p is sufficiently high gives us a tunable parameter– we can specify a threshold on p for a fixed recall k. In this way, we can also compare learned models against each other by identifying the precision achieved at a recall of k.

5.2.1 Defining Labels

We seek to build a model that can predict when to autocomplete, and if so, the clinical concept type of what follows. Given a set Γ of clinical concept types, our model S is predicting a distribution over $|\Gamma| + 1$ states: don't autocomplete, or autocomplete for each concept type in Γ. In our case, $\Gamma = 4$ and we predict over the following classes: `[NO AUTOCOMPLETE, AUTOCOMPLETE CONDITION, AUTOCOMPLETE SYMPTOM, AUTOCOMPLETE MED, AUTOCOMPLETE LAB]`. We can train this as a standard multiclass classification task with a cross-entropy loss.

5.2.2 Featurizing Text

Text data is inherently sequential, but we can featurize it in a variety of ways. We first tokenize the text by lowercasing it and splitting on whitespace. We also treat any phrase in our ontology of clinical concepts as a single token in order to easily encode compound phrases that carry a distinct meaning. Any numerical word is tokenized with a `NUM` token and split on. As an example, the text `pt p/w hypertension, coronary artery disease, on 20mg lasix` is tokenized as `[pt, p/w, hypertension, ',', coronary_artery_disease, ',', on, NUM, mg, lasix]`. We also experimented with further condensing the text to map any condition, symptom, medication, or lab to a single token representing each concept type, but this condensed tokenization scheme did not work better in practice.

Once the text has been tokenized, it must be encoded into input for our model.

Each token is featurized in two ways: a bag-of-words representation, or a word2vec embedding. In the bag-of-words formulation, tokens are converted into a sparse representation, but each feature vector's length is the number of unique tokens, which is over 200,000. Tokens can also be embedded via a word2vec representation, which we can train in an unsupervised way on a corpus of ED notes using the methodology in [80]. Word2vec representations of text, much like many embedding schemes, can align closely related groups of concepts in the embedding space. We compare bag-of-words representations of text against word2vec representations for some of our simpler models below, and unsurprisingly find that word2vec embeddings work better in practice. As a qualitative check, selected tokens and their nearest-neighbors in the trained word2vec embedding space are listed in Table 5.1. Distance is measured using cosine similarity. Formally, the cosine similarity between two nonzero vectors $\mathbf{A}, \mathbf{B}$ is defined as the cosine of the angle between them, or $\mathbf{A} \cdot \mathbf{B}/(||\mathbf{A}||\ ||\mathbf{B}||)$.

5.2.3 Dataset Generation

To generate our dataset to train S, we first identify all clinical concepts mentioned in the ED note using a trie-based match as in Section 4.2.1. We limit this search to the early HPI sections of ED notes on file, because our copy of the clinical notes are early drafts, and so later sections are not guaranteed to be fully-written.

We do not distinguish between positive and negative clinical concepts, as the token `no` can often be a powerful predictor of scope because it is often followed by a clinical concept. For a clinical concept c that appears at index i of a list of tokens W, we create a datapoint $(X, y) = (W[:i], c_{type})$. To generate negative samples, we select tokens preceding a word that is not a clinical concept uniformly at random, but throw away datapoints that aim to predict punctuation. As an example, if we were using the tokens `patient, has, diabetes` in our prediction and the following token was a comma, we would throw out this sample. In practice, we would never predict whether the user is typing punctuation and would trigger a prediction on spaces between tokens.

Hypertension	Warfarin	breast_cancer
Diabetes	pradaxa	lung_cancer
Hyperlipidemia	coumadin	melanoma
Congestive_heart_failure	dabigatran	melanoma
type_2_diabetes	lovenox	rcc
htn	xarelto	nsclc
hypercholesterolemia	anticoagulant	colon_cancer
hypothyroidism	plavix	prostate_cancer
diastolic_heart_failure	rivaroxaban	ovarian_cancer
chronic_kidney_disease	anticoag	renal_cell_carcinoma
mitral_valve_prolapse	anticoagulation	multiple_myeloma
history	alcohol	cough
hx	vodka	dry_cough
w/hx	alcohol_use	nonproductive_cough
pmhx	drinking_alcohol	non-productive_cough
h/o	consumed	non_productive_cough
diagnosis	rum	+cough
pmh	etoh	sputum
hisotry	drugs	sore_throat
w/history	etoh_use	night_sweats
histroy	alcoholic	nasal_congestion
w/pmh	listerine	rhinorrhea

Table 5.1: Word2Vec embedding quality in learned autocomplete triggering: shows the top ten closest tokens to each phrase, measured using cosine similarity.

Our final dataset consists of around 14 million samples. 13% of these samples are "positive" in that for a given datapoint (X, y), $y \neq$ `NO AUTOCOMPLETE` and the autocomplete scope is on. Of the positive samples, 66% are symptoms, 22% are diseases, 8% are medications, and 4% are labs.

5.3 Learned Triggers: Modeling

To satisfy an inference time of sub-100ms, model architectures have to be small. For this reason, we limit our architecture search to simple linear model classes such as logistic regression, as well as shallow neural networks that handle sequential data such as recurrent neural networks (RNNs) and one-dimension convolutional neural networks (CNNs). Because $\Gamma = 4$ and we only seek to predict a distribution over five classes, we find that even with these architecture constraints, our model works well in practice.

5.3.1 How much local structure do we need?

The inference time of our model is heavily dependent not only on its architecture but on the size of the input feature space. This is determined not only by how we encode words, but also by the amount of local structure we need to build a good predictor. Put simply, if a user is editing a text at some position i, how much text before and after i do we use in our model?

First, note that we cannot reliably use information after i just based on the nature of our problem. A doctor will typically craft a note by appending tokens to those that are also documented, which means there is nothing documented to the right of the cursor. This means that our local context must be a look-back model– we can only use words $W[:i]$. We must then determine how far the lookback context extends, of which there are two primary categories.

The first is to use a *persistent* lookback context. As the clinician types, they will

continue to append tokens to the text. We sequentially feed these tokens into a model that calculates a persistent hidden state (such as an RNN cell), and make a prediction about whether to autocomplete every time a new hidden state is calculated. While this framework can capture long-term text dependencies well because it could feasibly learn a representation of documentation style, it neglects the fact that a user can edit text that appears in the middle of a sentence. Because this breaks the assumption that documentation is append-only to the text, if a doctor were to edit a word at position i of the text, we would be forced to recompute the hidden state by feeding in all $w[:i]$, which is costly.

Alternatively, we could use a *fixed-window* lookback context. This means that if a user is editing text at a position i, we compute the last m tokens on-the-fly and use $w[i-m:i]$ as the input into our model. Our model does not then save any persistent hidden state and instead bases its predictions solely on the context provided at inference time.

We compare both persistent and fixed-window lookback contexts in the models below, but ultimately choose to use a fixed-window context because it saves computation time. This is a tradeoff– the persistent context models do perform slightly better on the prediction task, which we explore below.

5.3.2 Binary Prediction of Autocomplete Scope

In order to easily compare models, we first simplify our task to the binary prediction problem of autocomplete scope alone. This means we do not attempt to predict the concept type of the word that follows, and only output $P(\texttt{Show Autocomplete})$. This is a strictly easier problem than doing both autocomplete scope and type prediction as we are simply condensing Γ of our $\Gamma+1$ classes into a single class. We train two model types on this task: a logistic regression baseline, and RNNs using both a fixed-window and persistent lookback context.

Logistic Regression

In our logistic regression model, we experiment with the following architectures and featurization schemes:

1. **Lookback context:** Fixed lookback contexts with window sizes $m = 3 \cdots 10$. Logistic regression models do not support a variable-length persistent context because it doesn't store any hidden state.

2. **Text featurization:**

    - One-hot positional bag-of-words: For each token t in the lookback context, featurize with a a one-hot encoding of the vector. Concatenate these vectors to form a vector of length $m \times |V|$ where m is the windowsize of the lookback context and V is the vocabulary of ED notes.
    - Positional word2vec: For each token t in the lookback context, featurize t with its length-L word2vec embedding as per Section 5.2.2. Concatenate these vectors to form a vector of length $m \times L$.
    - Multihot bag-of-words: Tokenize all tokens in the lookback context with a single multihot vector, or as a bag-of-words representation. This forms a length-$|V|$ vector but loses all positional information.
    - One-hot positional bag-of-words with first character: Encode the context as per the one-hot positional bag-of-words scheme, but append a one-hot representation of the first *character* of the following word. We included this featurization to test whether knowing any information about the word the doctor is about to type (such as the first letter) can greatly improve our predictions.

All bag-of-words models were trained with heavy L_1 regularization, while models with word2vec featurization were L_2 regularized. Losses associated with positive labels were upweighted to counteract class imbalance (inversely proportional to the class frequencies in the input data).

Recurrent Neural Networks

Our RNN architecture uses a sequence of stacked RNN cells followed by a single fully-connected layer with sigmoid activations. Unidirectional, single-layer RNNs for the binary prediction task were trained using the following featurization schemes and hyperparameters:

1. **Lookback context:** fixed lookback contexts with window sizes $m = 3 \cdots 10$, as well as a persistent lookback context.

2. **Text featurization:** For each token t, we featurize it via a one-hot encoding as well as a word2vec embedding. Word2vec embeddings were not trained end-to-end in the binary prediction RNNs in order to fairly benchmark against logistic regression models, but are trained end-to-end in the final model.

3. **Hidden state:** 16, 32, 64, 128, 256.

4. **RNN cell type:** Long Short-Term Memory (LSTM) [50], Gated Recurrent Unit (GRU) [22], and SimpleRNN cells.

Knowledge Distillation

Large neural networks can have long inference times. However, these networks obviously have a greater number of parameters and can thus model a richer space of functions. One way to take advantage of this trade-off is to first train a large network on a task and then to *distill* its knowledge into a smaller network.

We use the framework proposed by [49]. Consider a large teacher model and a smaller student model. We first train the teacher model on our autocomplete scope task, and then train the student model using a composite loss $\ell = \alpha\ell_d + (1 - \alpha)\ell_s$. The first loss ℓ_d represents the *distillation loss*, which is typically the Kullback-Leibler (KL) divergence between the discrete distributions of predictions from the student and teacher models. ℓ_s, or the student loss, is the actual cross-entropy loss between

the predictions of the student model and the true ground-truth hard labels. α is a hyperparameter to learn an appropriate weight to attach to the distillation loss.

In our binary autocomplete scope prediction, this lends itself to an elegant training scheme. Let p and q be the estimated probabilities from the student and teacher models respectively (on a single sample), with some true ground-truth label r. Then ℓ simplifies to $(\alpha r+(1-\alpha)q)\log p+(1-(\alpha r+(1-\alpha)q))\log(1-p)+C$ where C is some constant. In other words, we can create a new target probability $\hat{p} = (\alpha r + (1-\alpha)q)$ and minimize the cross-entropy loss between $\hat{p}$ and p.

We experiment with two ways of parameterizing the student model S: first with a small RNN, and then with a simple linear model. In the latter, this amounts to training a linear regression to predict the logits (pre-sigmoid) of the these target probabilities. We then recover a classifier similar to logistic regression by applying a sigmoid layer to the predicted logits (subject to some distributional assumptions). We fix T to be our best performing fixed-window RNN (hidden size of 128, lookback window of $m = 8$ tokens), which achieves an AUC of 0.9 and a precision of 43% at a recall of 80%.

We find that the autocomplete scope problem is simple enough that our knowledge distilled models do not strongly outperform small models that are trained directly. A next step might be to try a more nuanced way of creating ℓ, e.g. via trust regularization, which modifies α over the course of training. We did not experiment with other network compression techniques such as weight pruning of low-rank factorization of weight matrices because even our "large" teacher networks are fairly compact.

Performance

We compare our binary scope prediction models with two metrics: AUC, to gauge threshold-independent performance that isn't sensitive to class balance; and precision at a fixed recall. We set recall to be 80% based on conversations with physicians and our goal to capture at least 80% of documented clinical concepts, but note that this

Model	AUC ↑	P @R=0.8 ↑
Rule-based	N/A	0.07
Logistic Regression		
One-hot positional BoW embeddings, $m = 5$	0.851	0.15
Word2vec positional embeddings, $m = 7$	0.865	0.27
Multihot BoW embeddings, $m = 3$	0.745	0.10
One-hot positional BoW embeddings plus first char, $m = 5$	0.860	0.12
RNN		
LSTM, One-hot BoW embeddings, $m = 10, h = 16$	0.880	0.30
LSTM, Word2vec embeddings, $m = 7, h = 16$	0.883	0.35
LSTM, Word2vec embeddings, $m = 8, h = 128$	0.900	0.43
LSTM, Word2vec embeddings, $m = \texttt{PERSISTENT}, h = 128$	**0.916**	**0.46**
Knowledge Distillation		
Linear model, Word2vec embeddings, $\alpha = 0.1$	0.85	0.33
GRU, Word2vec embeddings, $m = 8, \alpha = 0.6, h = 16$	0.911	0.42

Table 5.2: Performance of learned models for binary autocomplete scope prediction, as measured by AUC and by precision at a recall of 80%. BoW = bag-of-words, m represents the lookback context window size, h is the hidden state size of the RNN model

can be easily tuned with the decision threshold of the model. We also benchmark against a rule-based approach: We rank common unigrams and bigrams that occur before clinical concepts by frequency, and trigger autocomplete if the cursor is proceeded by one of the 2,000 most frequent unigram/bigram pairs. This achieves a recall of roughly 80% but is overfits to the training text.

A selection of models and their performance as shown in Table 5.2. Word2vec embeddings outperform bag-of-word embeddings, as expected, because we can exploit closely related terms in the embedding space. RNN models outperform logistic regression models, likely because they are nonlinear and a more complex model class. While the best-performing model is a LSTM trained with a persistent lookback context, we find that the latency and infrastructural challenging of maintaining a persistent hidden state removes the incremental performance gain, and instead opt for a fixed-window model.

5.3.3 Autocomplete Scope and Type Prediction

In the case of simultaneously predicting autocomplete scope and type from local context, given a set of clinical concept types Γ, our model outputs a discrete distribution over $|\Gamma| + 1$ classes– no autocomplete, as well as autocomplete t for each concept type t. In our setting, Γ is `[ CONDITION, SYMPTOM, LAB, MED ]` and there are thus five classes to predict over.

Developing a Loss Function

The standard way to measure loss in this framework is to use the cross-entropy across all five classes. However, we make the following observations. First, our binary scope prediction needs to be well calibrated because we threshold on $P(\texttt{NO AUTOCOMPLETE})$ to determine when to show the dropdown. On the other hand, the logits themselves for classes 1 through 5 do not matter– we only use their relative ranks to determine how to stack our rankings for each clinical concept type.

Second, in practice, training models with the cross-entropy objective yields peaky output probability distributions. This makes it difficult to establish a tunable threshold $P(\texttt{NO AUTOCOMPLETE})$ to achieve a particular precision/recall because the values are concentrated around 0 or 1.

To mitigate this, we construct a new way of measuring loss. Given model output probabilities $\mathbf{p} = p_0, p_1, \cdots, p_{|\Gamma|+1}$ from S and a ground-truth label γ^*,

$$\ell = \beta \cdot \text{xent}(p_0, [[\gamma^* = 0]]) + (1 - \beta) \cdot [[\gamma^* \neq 0]]\text{xent}(\mathbf{p}_{[\mathbf{1}:]}, \gamma^*), \tag{5.1}$$

where the cross-entropy function $\text{xent}(p, y)$ is defined by $\text{xent}(p, y) = -\sum_i y_i \log(p_i)$. Thus, we use a weighted combination of the binary cross entropy of our scope prediction task as well as a four-class cross entropy to incorporate the autocomplete type prediction task if applicable. Given our linear baselines did not rival performance of our RNN architectures for the strictly simpler task of binary scope prediction, we

limit our architecture search to RNNs and CNNs as discussed in the next section.

Note that because we consider neural network models like RNNs and CNNs that are optimized with stochastic gradient descent rather than a closed-form solution as per logistic regression, we train our input Word2vec embeddings end-to-end with our model. We experimented with different Word2vec embedding size (smaller embedding sizes obviously indicate fewer model parameters) and report the best architectures below.

Recurrent Neural Networks

Unidirectional, single-layer RNNs for the scope and type prediction task were trained using the following featurization schemes and hyperparameters:

1. **Lookback context:** fixed lookback contexts with window sizes $m = 4, 6, 8$.
2. **Text featurization:** For each token t, we featurize it via its length-h Word2vec embedding, which are trained end-to-end in the model.
3. **Hidden state:** 8, 16, 32, 64 (limit to smaller architectures).
4. **RNN cell type:** Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Simple RNN cells.

Convolutional Neural Networks

RNNs are inherently sequential as they calculate a hidden state that is then combined with a new input to make predictions. In contrast, CNNs do not do this– they instead perform efficient convolutions with learned filters on a single input matrix, which means they can be parallelized and are faster than RNNs to both train and perform inference with. We do not feed input tokens one-by-one into CNNs and instead feed in a one-dimensional vector of tokens to make a prediction.

While originally designed to handle image data for computer vision tasks, CNNs do have a semantic interpretation in our context. In the one-dimensional case, the learned

filters can identify "trigger words" that act as strong indicators of the autocomplete type. Filters of different sizes can equivalently learn "triger phrases".

Our CNN architecture is as follows:

1. Embedding layer to convert input tokens into their learned word2vec embeddings of length d_e. The number of tokens is determined by the length m of our lookback window.

2. 1-D Convolution layer with a kernel size s and a stride size of 1.

3. A fully connected layer which mean pools the output vectors from the previous layer, using learned coefficients.

4. Rectified Linear Unit (ReLU) activation, defined element-wise by the function $\text{ReLU}(x) = \max(x, 0)$.

5. A fully connected layer with $|\Gamma| + 1$ outputs, with a softmax activation, defined element-wise for a vector input z by $\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$, to convert the real-vector-valued output of the previous layer to a discrete probability distribution.

We optimize over hyperparameters d_e, k, s as well as β in our loss function.

5.4 Performance Results

5.4.1 Scope and Type Detection

Performance for our autocomplete scope and type prediction task is measured with two metrics: precision at a recall of 80% for our binary prediction task, as well as the mean reciprocal rank of our concept type rankings to gauge the efficacy of our type prediction. Results for the best performing RNN and CNN are shown in Table 5.3. While the RNN model has slightly higher precision at a fixed recall, the incremental gains from this are offset by its costly inference time (3 $\times$ slower than

Model	P @R=0.8 ↑	MRR ↑	MIL ↓
Rulebased	0.07	0.72	0.2 ms
RNN GRU $m = 4, d_e = 16, h = 64, \beta = 0.9$	0.45	0.89	54.9 ms
CNN $m = 6, d_e = 16, s = 4, \beta = 0.9$	0.43	0.90	17.1 ms

Table 5.3: Performance of learned models for autocomplete scope and concept type prediction, measured by precision at a fixed recall, mean reciprocal rank (MRR), and mean inference latency (MIL). In practice, the CNN model is deployed.

a CNN). Overall, the precision-recall curves of both the best-performing RNN and CNN models are similar, as can be seen in Figure 5-1. Ultimately, we deploy a CNN model. We further break down performance for the CNN model by concept type in Table 5.4. Our model has high MRR when predicting when *not* to autocomplete, as well as when to show the symptom autocomplete. This is due to both our training procedure, which favors the scope prediction over type prediction, as well as the fact that symptoms have common trigger words preceding them such as `presents with` or `complains of`. In contrast, MRR for labs is the lowest by a wide margin.

In Table 5.3, we report the *mean inference latency* (MIL), which is the average inference latency for a model in milliseconds (preparing text for the model and a single inference call). These times are measured on a single machine running Google Chrome with WebGL-backed acceleration, and can be subject to change on different machines. That said, their relative speeds does give a notion of comparative latencies. They are not representative of the system end-to-end latency which includes finding the user's cursor position from the underlying editor framework, parsing model output, and rendering the autocomplete dropdown itself. On our live-deployed system, the CNN model listed below averages approximately 18-19ms latency, which is very close to the refresh rate of the screen and is perceived as instantaneous to the user.

5.4.2 Overall

In an evaluation of our retrospective data, our CNN joint scope/type prediction has a 92% accuracy of showing the autocomplete dropdown before a clinical concept,

Concept Type	MRR ↑
No Autocomplete	0.91
Autocomplete Condition	0.76
Autocomplete Symptom	0.84
Autocomplete Medication	0.75
Autocomplete Lab	0.39

Table 5.4: Autocomplete scope and type performance for the best-performing CNN model, broken down by class.

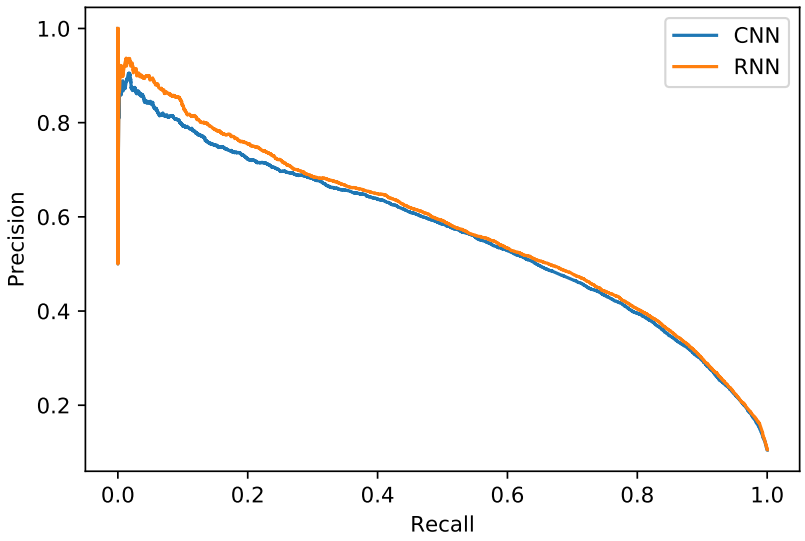


Figure 5-1: Precision-Recall curve for best learned scope/type prediction models, as in Table 5.3.

without the need for a manual trigger or a postcorrection. In 85% of cases, the concept type of the desired concept was ranked first. Overall, we reduce keystroke burden from 8.30 to 2.63 (a 68% decrease).

5.5 Summary

The contextual autocomplete ranking models we have outlined harnesses the power of machine learning to encode information about medical contexts, and then uses this to suggest terms to document to clinicians. Medical professionals who utilize this tool can not only document terms more easily and save valuable time to interact directly with patients, but also can create clean annotations of clinical text in a novel manner. These annotations can be used to provide disambiguation between overloaded terms, clarify associations between medical concepts, and generate large-scale EHR datasets for future innovation. All of the medical ontologies built for this work map to UMLS, making our contextual autocompletion tool translatable to other clinical centers with minimal modification.

In this chapter, we focused on developing autocomplete scope models to predict when to display the autocomplete dropdown, and whether we could use the local structure within notes to guess clinical concept types.

5.5.1 Future Direction: Integrating Semantic Modifiers

The contextual autocomplete tool in this thesis relies on four concept types –conditions, symptoms, medications, and laboratory tests– to create a thorough vocabulary of UMLS-mapped clinical concepts. However, this is by no means an exhaustive representation of clinical language. In particular, we need a better way of integrating key semantic modifiers in our framework.

We define a *semantic modifier* to a clinical concept as any sequence of words that directly alter the meaning or medical relevance of a concept. `No` is a semantic modifier

of `fever` in the phrase `No fever` because it changes whether the symptom was indeed present in the patient. On the other hand, `coumadin` is not a semantic modifier in `afib (on coumadin)` because while it directly relates to `afib`, it does not impact the semantics of the concept.

There are four types of semantic modifiers that we seek to capture. We list them below along with our current strategy of integrating each into our system.

Negations

Doctors often document the absence of concepts (e.g. `no fever`) to aid in a differential diagnosis. In the ED, this is usually limited to documenting "pertinent positive (present) and negative (absent)" symptoms. We support the entry of negative symptoms of the form `no <SYMPTOM>`. In order to detect negative symptoms in a list of concepts such as `no fever, nausea, or chill`, we currently use NegEx, which is a rule-based approach [19]. Negated terms are displayed differently from positive ones to distinguish them in the UI. A future direction to improve data entry in the tool would be to incorporate a learned model for negation detection rather than a rule-based one. This might be possible even with minimal training data by using extending NegEx with kernel methods [109] or dependency parsing [79].

In addition, once negations are better structured and capture with the UI, this can be used to improve our autocomplete ranking models to suggest negative symptoms and conditions to document.

Adjectives

Clinical concepts are often modifier with *adjectives* that clarify their semantic meaning. Adjectives can be used to specify spatial orientation (`right-sided chest pressure`), clarify body systems (`abdominal tenderness`), indicate severity (`severe bleeding`), describe a quantitative relationship (`elevated cholesterol`), or even stipulate temporal relations (`intermittent pain`).

Deciding when and how to modify a clinical concept with a descriptive adjective is purely subject to physician preference and training. In our pilot system, we do not attempt to predict `<modifier> <concept>` pairs in our ranking algorithms R.

However, we do support the post-hoc attachment of modifiers to clinical concepts during data entry. As an example, if a doctor types `right-sided chest pain` and `chest pain` is tagged as a clinical concept, we greedily look for adjectives that precede the concept to attach to it. In this way, as we develop a large corpus of annotated clinical text, we can robustly learn allowable adjective modifiers to attach to each clinical concept.

To develop our ontology of acceptable adjective modifiers, we looked for phrases that appeared directly before clinical concepts in the ED notes on file, restricting to phrases that were denoted as a Qualitative Concept, Temporal Concept, or Spatial Concept in UMLS. There are 255 adjective modifiers in our current ontology.

Third-Party Attribution

Clinicians might also type a term that refers to someone other than the patient, such as `family history of diabetes in mother`. This is an example of a *third-party attribution* modifier to `diabetes`, because it indicates that the clinical concept in question should not be assigned to the patient. In the ED, third-party attribution modifiers tend to occur only within the Family History section of the note where the doctor describes diseases and conditions associated with relatives. We examined two different ways of capturing third-party attributions in Family History sections.

The first is to rethink how to capture data within Family History sections. All that is being recorded is a bipartite matching between family members and conditions. This can be captured in a table or a more structured input form as opposed to free-text, which can then be programmatically parsed. BIDMC already has a rudimentary version of this system and it is not used by most doctors because free-text is more convenient.

The second is to use an algorithm (either rule-based or learned) to identify family members and third parties present in the text, and then attempt to match clinical concepts to those subjects. This can be abstracted as a NLP relation extraction task– given a piece of text, first extract entities (family members, clinical concepts) and then determine their relationship.

There is a dearth of literature that examines relation extraction in the context of clinical text, let alone for family history detection. In the 2018 BioCreative/OHNLP challenge, which asked participants to perform entity recognition and relation extraction for the purpose of family history detection, many of the submissions used a purely rule-based approach, and the best-performing model only achieved an F1 score of 0.57 [14, 46, 66]. Performance is much too low to be deployed in a live hospital setting. This is likely because the OHNLP challenge lacked sufficient labeled data to learn relations robustly; in contrast, state-of-the-art relation extraction models in the general domain are usually deep neural networks that ingest vast amounts of data [126]. In recent years, there has been a focus on using self-attention for relation extraction on top of common neural architectures such as convolution neural networks and transformers [127, 71]. However, we do not have the data to train these models to perform relation extraction in a supervised way.

To some extent, family history extraction is similar to coreference resolution in classical NLP wherein one tries to find all expressions that refer to the same entity in text. There is some preliminary evidence that contextual language models like BERT encode syntax through self-attention, and may be able to be used for coreference resolution and subject-verb agreement [26, 43]. Still, there is contention in the field whether self-attention weights can be reliably used, due to the overparameterization of BERT [61] and redundancy of attention mechanisms [53].

Assuming that attention-based models do encode some semantic representation of text, we can use the following semi-supervised algorithm to perform family history extraction.

1. Train a BERT model on all OMR notes $\mathcal{H}$. This is unsupervised because BERT

uses masked-language modeling as its training task. In practice, we use the same architecture as in [30] but define a custom vocabulary of clinical terms. To generate a list of meaningful subwords and tokens, we use the SentencePiece byte-pair encoding method [62].

2. Use a rule-based approach (RegExes with minor post-processing) to segment Family History sections from discharge summaries on OMR notes. 67% of patients in our dataset had such a section in their OMR notes.

3. Tokenize each Family History section into sentences. Extract conditions mentioned in the text using a trie of UMLS terms. Extract family members mentioned in the text with string matching on `mother`, `grandfather`, etc.

4. Generate BERT attention weights for each sentence. We chose to max pool over the five best-performing attention heads (evaluated on a small manually labeled validation set of 30-40 relations).

5. Match each condition to the (singular) family member that it attends most to. If there is no family member present in the text, assign the condition as "generally" relevant.

In some cases, this algorithm works well: running it on `History of acute leukemia in older brother who has passes way, early breast cancer in sister, smoking-associated lung cancer in another sister` results in `acute leukemia` being attributed to a `brother` and all other concepts to a `sister`. However, if a condition should be attributed to multiple family members, it fails: hypertension in her mother and sister attributes `hypertension` to `sister` but not `mother`. More alarmingly, the inherent shorthand and messiness of clinical text can sometimes result in erroneous pairings– running this algorithm on `mother with hypertension, diabetes in father` results in both concepts being attributed to `mother` and none to `father`.

Clearly, an unsupervised attention-based matching algorithm works on simple cases and sentence structure but fails on insidious and difficult cases. However, with the

curation of larger-scale datasets for relation extraction in clinical text, it may be possible to fine-tune this algorithm by training it in a supervised manner. This is a proof of concept and an initial foray into automatically detecting third-party attribution modifiers via self-attention.

Hedging

The last type of semantic modifier we note is *hedging*, which is when a physician might indicate uncertainty about a claim, e.g. `patient may have Lyme disease`. Out-of-the-box hedging detectors are usually rule-based or cater to specific note-types such as radiology notes [87] or biomedical articles [1]. As such, we did not explore hedging for the first iteration of the system.

Chapter 6

Patient Record Summarization from Unstructured Text

Having discussed the data entry portion of the tool, we now move on to the contextual information retrieval that is enabled as a result of tagging clinical concepts. In this chapter, we focus on retrieving unstructured textual data from a patient's prior medical record (OMR). In essence, we are attempting to build an *extractive* and *indicative* summarization of a patient's disease history from a patient's OMR.

In order to gather a complete and thorough history prior to treating a patient, physicians will first read through a patient's OMR and manually search for sections of a note that are relevant to the current presentation. As an example, given a patient complaining of dyspnea, a prior history of chronic lung disease could greatly change the course of treatment. Because the vast majority of a patient's medical record lies within unstructured clinical notes, doctors are forced to search through these medical records without any guidance, which is tedious and time-consuming. By suggesting snippets from the OMR, we can paint a disease-specific summary of a patient's history, which could then alleviate work for the clinician.

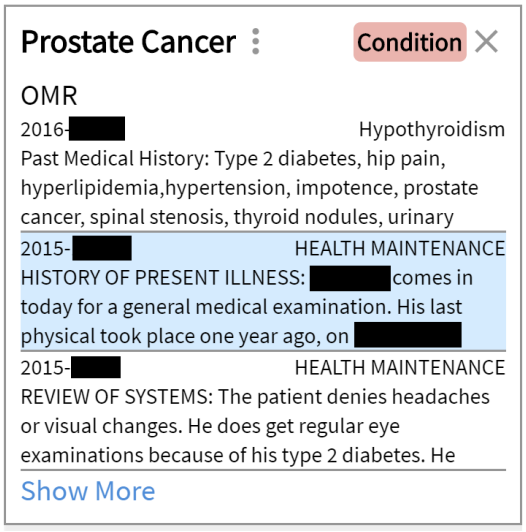
However, we do not want to completely replace this step. Relying solely on a learned model to surface relevant snippets from a medical record can be dangerous, because

it injects algorithmic bias into critical clinical decisions. Instead, once a doctor tags a concept using contextual autocomplete, we seek to surface snippets related to that particular concept, which we then populate on the sidebar so the doctor can click to further explore. A concrete example of this is shown below in Figure 6-1. This summarization approach, in some senses, a generalization of electronic medical record phenotyping– we are assigning snippets labels that indicate how relevant to each concept they are. We first formalize the problem of OMR snippetization and then discuss different machine learning methodologies to solve it, inspired by literature on EHR phenotyping [48] and clinical summarization [91].

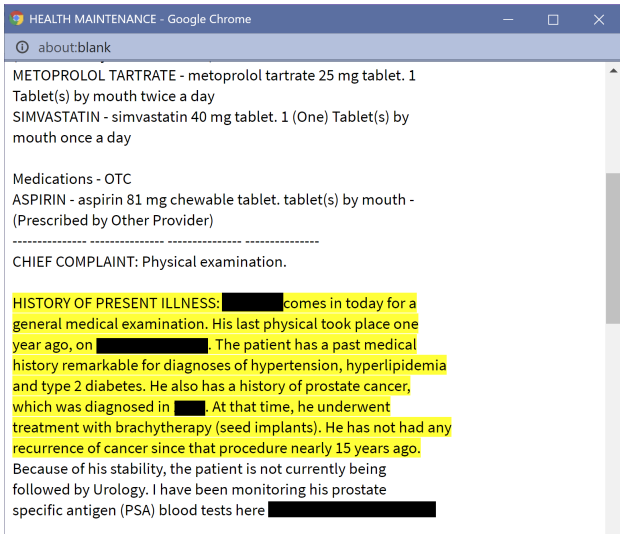
6.1 Formalizing OMR Snippetization

At the level of an individual patient, we split the full record of all OMR notes into a set *snippets* $\mathcal{S}$. Each $S_i \in \mathcal{S}$ consists of an ordered list of words, and our goal is to determine the *relevance* of a snippet to a certain condition c by learning a function $r(S_i, c) \in \mathbb{R}$ that assigns a score to any snippet-condition pair. If we can learn such a function that reflects a reasonable measure of relevance, then when a clinician mentions a condition c, we can surface the most relevant snippet $\text{argmax}_{S_i \in \mathcal{S}} r(S_i, c)$, or several top-ranked snippets.

A clear baseline that is used in BIDMC's current system is a direct keyword search. That is, we assign a binary relevance score $[[c \in S_i]]$ to a (snippet, condition) pair (S_i, c) reflecting if the condition c is directly mentioned in the snippet text itself, potentially ranking in reverse chronological order so newer snippets are surfaced first. There are several clear augmentations to this method, such as checking if any known synonym of c occurs in S_i. However, this basic technique does not allow us to show snippets that contain non-obvious related information, such as a relevant medication or procedure. Moreover, it snippetizes the OMR at a note-level rather than a sentence-level, forcing doctors to read an entire note to find a sentence or two of salient information. Thus, we seek to define more robust functions $r(S_i, c)$ that better



Card with snippets



Full note exploration

Figure 6-1: OMR Snippetization UI. Left: Snippets from the patient's OMR that are related to a particular condition are listed on the condition's card. Right: Clicking on any of these snippets then opens a pop-up screen that depicts the note in full, with the original snippet highlighted.

align with clinical intuition and needs.

Ideally, we can learn r by using a labeled dataset that can serve as a proxy of relevance. As an example, click-through data showing what OMR notes a physician browses for each patient might give us a lens into what is relevant. For this pilot, we did not have access to any data source like this. Instead, we only have access to drafts of the clinical ED note, as well as finalized, full-length OMR notes. Thus, we rely on a variety of un- and semi-supervised techniques and external knowledge bases to guide our decision of relevance. We hope that our current tool can help collect more direct measures of snippet relevance. We also build an OMR relevance annotation tool to curate a large-scale dataset of relevance, should the need arise. The annotation tool is described in Appendix A.3.

6.2 Dividing Notes into Snippets

We do not want to force a clinician to read through an entire note in order to find a single section that is relevant. Instead, we look for short phrases that are relevant in the current medical context and act as summaries of the desired information– this requires segmenting a note into its constituent snippets. One way of doing this is to use natural breakpoints in the text, such as splitting by note section. However, because OMR notes can potentially date back decades and contain a variety of note types and clinical specialties, there is no consensus on note structure. Segmenting a note by section is thus infeasible without many hand-crafted rules.

Doctors can always expand the segment of the note they read beyond the provided snippets, so we instead take a heuristic approach to segmentation. We first divide the note into coarse paragraphs by splitting on double newline characters. We then further split each paragraph into sentences. Finally, we generate our candidate snippets by taking rolling windows over groups of 2-3 sentences, enforcing that snippets are around 50-150 words. Shorter snippets are coalesced into larger ones to have enough text to be predictive, and long snippets (over 150 words) are split to ensure that snippets

were uniform and did not cover too much ground. An example of candidate snippets generated for a note is shown in Figure A-3 of the Appendix.

Future iterations of the snippetization algorithm can make use of section headers to ensure that snippets do not span multiple sections or to exclude certain sections like Family History.

6.3 Measuring Snippet Relevance

We now explore multiple ways of measuring snippet relevance. In practice, we use a data-driven heuristic that extends keyword search to closely-related concepts of the query, as described next in Section 6.3.1. This is an extendable and interpretable model. We also experimented with two learned schemes for performing snippetization (topic modelling, and anchor-and-learn phenotyping) which we touch upon as well.

6.3.1 Advanced Keyword Search

The exact-string-match keyword search baseline is easy to justify because it is consistent: it returns notes in reverse chronological order so doctors understand when information is out of date, and the behavior is deterministic. Its primary drawback is that it is inherently restrictive– searching `diabetes` should also pick up on notes using acronyms (`dm`), related medications (`insulin`) and other salient information. In the advanced keyword search paradigm, we seek to fix this by expanding the set of acceptable keywords that trigger a match for a particular concept. Thus, instead of using the relevance score $[[c \in S_i]]$ for a concept c and a snippet S_i, we define a set of acceptable keywords K_c and set $r(S_i, c) = [[|K_c \cap S_i| > 0]]$, or that there is at least one keyword within K_c that also occurs in S_i.

We posit that keywords for a term will always be a clinical concept that falls into one of the following categories:

- Conditions: any chronic disease or diagnosis. These form the bulk of our

keywords– as an example, snippets that are relevant to AIDS might contain the keywords `AIDS` and `acquired immunodeficiency syndrome` but also common coinfections like `hepatitis` and `pneuomnia`.

- Medications: any drug that would treat the condition. `Lamivudine`, for example, is a common antiretroviral therapy that treats AIDS.

- Procedures: any recurring treatment or prior surgery that is relevant in the context of the disease. In the case of AIDS, any surgical history is likely relevant, but `appendectomy` might be an acceptable keyword only if the query concept is related to abdominal pain. We assume access to an ontology of procedures in this chapter, but describe the data curation process in the next.

We note that this list does not include symptoms and labs. While labs are crucial to understanding disease prognosis, they also exist as structured data and can be visualized in our system without having to extract from unstructured text, as elaborated on in Section 7.2. On the other hand, acute symptoms can occur due to a variety of conditions and are hard to directly attribute to a clear source. Thus, we exclude symptoms from our keywords. Ideally, we can rely on structured knowledge bases to generate K_c for each condition c. This is especially easy for related medications to a condition. We use existing UMLS-mapped drug-disease indications from [108]. Any name (generic and brandname) of a drug that treats c is added to K_c. We discuss these drug-disease indications in further detail in Section 7.1.

While knowledge graphs have been widely used in healthcare to learn relationships between clinical concepts and to aid predictive modeling [102, 23, 75], they are often disease-specific or too coarse-grained to use for our purposes. Indeed, we do not care about *all* related conditions/procedures to `diabetes`, but rather a specific and small subset that directly affect the disease prognosis like `diabetic retinopathy`, `hyperlipidemia`, etc.

We make use of an external sources of knowledge to seed our condition/procedure keywords, and then manually pare them down to what is relevant with clinical consul-

tation. We use pretrained low-dimensional embeddings for UMLS CUIs corresponding to conditions and procedures in our ontologies [13], and for a concept c, seed its condition/procedure keywords with its nearest neighbors using cosine similarity with a manually tuned threshold of 0.7. We then the warm-start keywords to what is actually and directly clinically relevant.

In order to fill in the gaps with this method due to limitations with our ontologies as well as to corroborate our keywords with what physicians are actually documenting, we complement it with a more data-driven method. We extract clinical conditions in patients' ED notes, find snippets of corresponding OMR notes that mention the same condition, and then run a UMLS-based extraction to find mentioned UMLS concepts in these OMR notes, not limiting to concepts that are in our ontologies. Co-occurring conditions for a condition c were then ranked by the fraction of time they co-occur with c compared to the number of overall mentions– this downweights common concepts like `hypertension` that co-occur with almost every disease. With clinical consultation, we add any missed keywords for each concept. As an example, for the type 2 diabetes concept, we add `diabetic retinopathy` and `cerebrovascular accidents` as keywords, but exclude medications that are already captured with drug-disease indications like `insulin`, as well as common comorbidities or noise like `orthostatic hypotension` and `eye exam`.

In short, we instantiate a keyword set K_c for a condition c first with any synonyms of c, then with any drug that treats c, and finally any synonym of a condition/procedure UMLS concept caught by the two-pronged manual curation described above. Examples of these keyword sets for select concepts are shown in Table 6.1.

While it is difficult to conduct a quantitative evaluation of this advanced keyword search without ground-truth labels of relevance, we show case studies of our method compared to standard keyword search below in Table 6.2.

Condition: Type 2 Diabetes	
Direct synonyms	adult onset diabetes, t2dm, type 2 diabetes mellitus, niddm, ...
Medication Keywords	insulin, metformin, glucotrol, metaglip, novolin, ...
Embedding Neighbor Keywords	hyperlipidemia, hypercholesterolemia, prediabetes, impaired glucose tolerance, brittle diabetes, insulin-dependent diabetes
OMR Co-occurrence Keywords	diabetic retinopathy, cerebrovascular accidents
Condition: Asthma	
Direct synonyms	asthma, childhood asthma, exercise-induced asthma, bronchial asthma, allergic asthma, ...
Medication Keywords	albuterol, pseudoephedrine, salmeterol, fluticasone, beclomethasone, ...
Embedding Neighbor Keywords	rhinitis, allergic rhinitis, hay fever, rhinorrhea, chronic rhinitis, lung disease, tracheobronchomalacia, pneuomnectomy, tracheostomy, pneumothorax, pneumonia
OMR Co-occurrence Keywords	nebulizers
Condition: Migraines	
Direct synonyms	migraines, migraine headaches, complex migraine, complicated migraines, migraine syndrome
Medication Keywords	naratriptan, sumatriptan, zolmitriptan, eletriptan
Embedding Neighbor Keywords	hemiparesis, TIAs, CVAs, stroke, seizures, aphasia, epilepsy, ...
OMR Co-occurrence Keywords	photophobia, phonophobia

Table 6.1: Case Studies: Snippets surfaced with standard vs. advanced keyword search for a sample of conditions and patients.

Standard Keyword Search	Advanced Keyword Search
Condition: `Anxiety`	
September 10, 2013: His past medical history include pituitary tumor. He also has IBS and anxiety. January 8, 2013: Librium 5 mg twice a day for quite a few years to control anxiety. Also takes omeprazolae 10mg daily.	May 13, 2014: on Prescription Cabergoline (0.5 mg tablet), Librax with Clidinum (5mg capsule). September 10, 2013: His past medical history include pituitary tumor. He also has IBS and anxiety. January 8, 2013: Librium 5 mg twice a day for quite a few years to control anxiety. Also takes omeprazolae 10mg daily. December 26, 2008: Complains of insomnia and occasional panic attacks at night. Starting on low-dose chlordiazepoxide to treat.
Condition: `Osteoporosis`	
July 12, 2006: Trazadone 50 mg q h.s., and Miacalcin spray for osteoporosis. May 1, 2006: Current medications include lipitaor, coumadin, lasix, levoxyl, protonix, KCL, Diovan, Trazodone, and Miacalcin spray for osteoporosis. Allergic to penicillin, erythromycin, biaxin, and zithromax.	January 30, 2007: She has switched to Aredia by IV infusion from previous Miacalcin spray. Die had an MI in Florida in 2004 and underwent emergency catherization. January 10, 2007: Miacalcin spray for osteoporosis, but history of multiple allergic reactions to medications including (most importantly) several reactions to beta blockers with increasing asthma. July 12, 2006: Trazadone 50 mg q h.s., and Miacalcin spray for osteoporosis. May 1, 2006: Current medications include lipitaor, coumadin, lasix, levoxyl, protonix, KCL, Diovan, Trazodone, and Miacalcin spray for osteoporosis. Allergic to penicillin, erythromycin, biaxin, and zithromax.

Table 6.2: Case Studies: Snippets surfaced with standard vs. advanced keyword search for a sample of conditions and patients.

6.3.2 Latent Dirichlet Allocation and Topic Modelling

We now explore two learned mechanisms to find relevant snippets given a condition, and discuss why they are not used in practice. The first of these is topic modelling.

The classical paradigm to assign genres or labels to natural language is known as *topic modelling*. Topic models are statistical models that discover underlying topics to a collection of documents and are commonly used to identify latent semantic structure in a text corpus. In the case of finding relevant portions of OMR given a medical context, we can use topic models to classify snippets as representative of a given condition or disease. As an example, if one wanted to look for snippets related to the condition `diabetes`, we might assign topics to all of our candidate snippets and return those that are highly weighted towards `diabetes`. We explore variants of the latent Dirichlet allocation (LDA) topic model to determine the relevance of a snippet to a particular concept. While topics were too coarse and uninterpretable for actual deployment in the tool, these act as valuable snippetization baselines.

Traditional LDA

The prototypical topic model is Latent Dirichlet Allocation (LDA), which has been widely used and extended [15]. It is a probabilistic graphical model that (in the case of text data), given a set of documents that are comprised of words, posits that each document is a mixture of a small number of topics and that the words in a document are drawn from distributions parameterized by these topics. Note that documents can exhibit multiple topics. We fix the number of topics in advance and use a generative process to describe how documents are created– first, we randomly choose a distribution over topics, and then for each word in the document, we randomly choose both a topic from the document's distribution over topics as well as a word from the corresponding topic. Words are generated independently of other words, and as such, a traditional LDA treats a document as a unigram bag-of-words.

Formally, consider a set of D documents $d_1, d_2, \cdots d_D$ that each are defined by some

mixture of K topics. Each document is made up from an unordered sequence of words chosen from a total vocabulary of size V. We notate the number of words in document d_i by N_i. Let the set of vectors $\beta_1, \beta_2 \cdots \beta_K$ represent prior distributions over words associated with each topic, with the vector β_k the prior distribution for topic k. Likewise, let the vector α represent the prior for the weight of each topic in any document, with the scalar value α_k representing the prior weight of topic k.

We define a generative process as follows. First, for each document d we draw topic proportions $\theta_d \sim p(\theta_d; \alpha) = \text{Dir}(\alpha)$, and for each topic k we draw a distribution $\phi_k \sim p(\phi_k; \beta_k) = \text{Dir}(\beta_k)$ over words for the topic. We draw a topic for each word of each document $z_{d,n} \sim p(z_{d,n}|\theta_d) = \text{Categorical}(\theta_d)$, where $z_{d,n}$ is the topic assignment for the nth word of document d. Finally, we draw the nth word of document d as $w_{d,n} \sim p(w_{d,n}|z_{d,n}, \phi_{1:K}) = \text{Categorical}(\phi_{z_{d,n}})$. Under this generative model, we can write the joint distribution of all variables (both hidden and observed) as follows:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{k=1}^{K} p(\phi_k; \beta_k) \prod_{d=1}^{D} p(\theta_d; \alpha) \prod_{n=1}^{N_d} p(z_{d,n}|\theta_d) p(w_{d,n}|z_{d,n}, \phi_{1:K}) \tag{6.1}$$

While the original paper [15] approximates this quantity via variational inference, one can also do so via Markov Chain Monte Carlo techniques, which we opt for. We use the Gibbs sampler provided by the MALLET software package [78]. We train a 100-topic LDA model on a random sample of OMR note snippets, using space-delimited tokens as words in our vocabulary.

Examples of the words generated by these topics are shown in Table 6.3. While we have prior knowledge about token similarity from domain expertise (`afib` and `aflutter` are relevant in almost identical medical contexts), we do not make use of this in our completely unsupervised LDA setup. In particular, the LDA algorithm proposed in [15] assumes a uniform prior over topics for each word in the vocabulary, whereas we might strive to seed each topic with correlated words. Nevertheless, from the perspective of snippetization we must map from LDA topics to clinical concepts. We do so directly by establishing a one-to-one correspondence between topics and

Topic Name	Topic Words
Skin	rash skin lesions face cream allergic back reaction apply topical neck scalp benadryl areas chest area dermatology itchy allergy examination
Cancer	cancer mass breast lesion biopsy radiation treatment disease lymph showed carcinoma scan tumor lung metastatic cell noted therapy lobe prostate
Hand	hand finger wrist fracture arm elbow injury splint distal shoulder thumb motion radial intact swelling joint forearm ulnar range fingers
Breast	breast exam pap cancer masses negative discharge smear lesions health exercise discussed sexual family mammogram screening year check age annual
OBGYN	pregnancy vaginal bleeding pelvic weeks obgyn fetal cyst lmp gyn pregnant ovarian urine ultrasound uterus blood prenatal discussed negative cervix
Cardiac	aortic artery cardiac stenosis valve procedure mild carotid surgery vascular mitral stent ventricular findings prior cad vein systolic lad femoral
?	physician chart visit note deaconess final attending social created initial systems reviewed department constitutional diagnosis procedures emergency entered version real
?	today fracture physical plan examination document needed weeks surgery prescription medication present illness post followup visit assessment incision status hip

Table 6.3: Sample Topics generated with traditional LDA on snippets of the OMR. Trained on $K = 100$ topics. Manually-chosen topic names are shown for medically-meaningful groupings.

concepts in the next section.

Labeled LDA

Traditional LDA is completely unsupervised in that each document's distribution of topics is unknown. We instead want to force each topic to represent a condition in our ontology. We also want to be able to automatically seed a given snippet with multiple conditions/topics– a phrase like `diabetes and hypertension` might be equally relevant to two different topics. In short, we want to generate LDA topics from multi-labeled corpora– a paradigm coined labeled LDA (LLDA) in [94].

Like traditional Latent Dirichlet Allocation, LLDA models each document as a mixture of underlying topics, generating each word from one topic. However, LLDA is supervised in that it constrains the topic model to use only topics that correspond to a document's observed label set. Formally, we label each document d with a set of topics expressed as a multi-hot vector ζ_d. We modify the generative process by first deterministically computing the vector $\alpha_d = \frac{\alpha \odot \zeta_d}{|\alpha \odot \zeta_d|_1}$, where $\odot$ represents element-wise multiplication. In this way, for each document d, we modify the prior over topics to be restricted only to the labeled topics, then normalize the prior distribution. The new generative process draws $\theta_d \sim p_{\text{labeled}}(\theta_d; \alpha, \zeta_d) = \text{Dir}(\alpha_d)$, and then proceeds identically to the case of traditional LDA.

To translate LLDA into our paradigm, we first define that each topic will correspond to a model relevance bucket as defined in Section 4.2.1. We use these coarser buckets instead of individual clinical concepts simply as a first-pass grouping. Each OMR snippet is then assigned a label corresponding to topic k if any concept in model relevance bucket k is mentioned in the snippet. While topics generated from training LLDA on snippets of the OMR are ostensibly more meaningful than their LDA counterparts, they are relatively coarse-grained and often capture unwanted comorbidities. As an example, the LLDA topic corresponding to the `diabetes` model relevance bucket places high mass on the word `lisinopril`, which is an antihypertensive medication. This is because patients are often diagnosed with both hypertension and diabetes, even though their disease histories are separate. Overall, the topics look reasonable but also noisy, as shown in Table 6.4– considering we are labelling at the coarse model relevancy bucket-level, LLDA would be difficult to cleanly extend to individual concepts.

6.3.3 Anchor-and-Learn

Instead of taking a generative approach to snippet relevance, we can instead use a discriminative one. For a given snippet S_i and condition c, we can estimate $r(S_i, c)$ as the *probability* S_i is relevant to c and rank by this soft quantity. Our choice is then

Topic Name	Topic Words
Hypertension	status intact bilaterally home activity gait functional head sit impaired htn sleep balance noted minutes unable rehab strength mobility motor
Diabetes	diabetes insulin needed times release units otc metformin medication capsules aspirin capsule lisinopril list pressure active prescription dose sugar weight
Arteriopathic diseases	artery impaired stenosis bypass carotid femoral grossly vascular graft stent claudication iliac common total shoulder lower angioplasty voice plavix pvd
Cardiomyopathies	cardiac cad disease artery coronary mild aortic cabg heart valve prior aspirin bid lad ventricular edema htn mitral cardiology catheterization
Depression/suicide	depression psychiatric reports abuse assessment include suicidal axis mood admission suicide states treatment ago ideation amp substance thought thoughts anxiety
Asthma	asthma albuterol needed cough inhaler hfa breath puffs sulfate shortness wheezing prednisone mcg aerosol exacerbation hours call advair regular doctor
Chest pain	chart test final deaconess emergency physician department ekg heart diagnosis rhythm neuro constitutional created systems stress ecg rate attending presenting
Hyperlipidemias	pressure hypertension heart cholesterol weight disease edema review aspirin rate exercise hyperlipidemia dear rhythm clear regular recent test months coronary

Table 6.4: Sample topics generated with LLDA on snippets of the OMR. Each topic corresponds to a model relevance bucket as defined in Section 4.2.1.

how to parameterize r– a straightforward framework is logistic regression.

However, we also need to define our notion of relevance in this setting. Inspired by literature on unsupervised clinical phenotyping, we turn to the *anchor-and-learn* paradigm proposed by [48]. Consider a set of documents $D = d_1, d_2, \cdots$. For each clinical concept of interest c, we establish certain textual *anchors* A_c that act as certain indicators that some document d has c as a phenotype– for example, $A_c = \{\texttt{atrial fibrillation}, \texttt{afib}, \texttt{af}, \texttt{a fib}\}$ for the concept "atrial fibrillation". Then, we extend the markers that correspond to concept c as follows: for each d, we create a new document d' that masks out any anchor of c from the text. We learn a mapping (typically a logistic regression model) to predict $[[|A_c \cap d'| > 0]]$, or the presence of any anchor in the masked document d'. Ideally, we pick up on predictive features of our phenotype; for atrial fibrillation, we might find that `coumadin` and other blood thinners are highly-weighted features.

We treat each of our concepts as distinct phenotypes and snippetize sections of OMR notes into individual documents. For a concept c, we set its anchors A_c as any acceptable synonym of the concept and train a logistic regression anchor-and-learn model to predict the presence of an anchor in masked document. To encourage the model to use related clinical concepts as predictive features, we preprocess the text to coalesce tokens that correspond to the same clinical concept. Like [48], we also detect concepts that occur within a negated scope using the NegEx algorithm from Section 4.2.1, and prepend them with `no_` to capture negative concepts. As a concrete example, the phrase `patient has a history of diabetes mellitus, but no fever or chills` would be preprocessed as `patient has a history of diabetes_mellitus but no_fever or no_chills`. We then tokenize the text with a binary multi-hot bag-of-words representation, and train a highly-L_1 regularized logistic regression model to encourage feature sparsity. A bag-of-words representation is natural for this application because it mimics how humans might determine the relevance of a snippet– by looking for words or phrases that match the desired query. Using this procedure, there are 126,714 unique tokens in our dataset.

Note that `atrial fibrillation` will not be a predictive feature for the atrial fibrillation concept because it is masked out of documents in our training set. To rectify this, we must modify our inference procedure. For a given condition concept c, let f_c be its corresponding anchor-and-learn logistic regression model. We first generate a set of candidate snippets $\mathcal{S}$. For each $S_i \in \mathcal{S}$, we first find whether any anchors of c are present in S_i, as well as the probability of the snippet being relevant if the text were masked. Thus, our relevance measure is

$$r(S_i, c) = [[|A_c \cap S_i'| > 0]] + f_c(S_i)$$

In this way, any snippet where an anchor for concept c is mentioned is automatically ranked above a snippet without the anchor present. The non-anchor tokens in the text determine its ordering, such that the presence of highly predictive tokens upweight the snippet's relevance. Top-weighted coefficients for a sample of models (each corresponding to a single condition concept) are shown in Table 6.5. On average, there are 607 nonzero coefficients per model (a sparsity of 0.4%).

We see that while these features seem more robust compared to those generated from LLDA, we still see many of the same problems– disease comorbidities for two relatively unrelated conditions (e.g. `diabetes` and `hypertension`) and meaningless words that are predictive (e.g. `pmh`). These problems are hard to mitigate without some amount of manual curation.

Although we can calculate the performance of our logistic regression model for each concept with a standard metric like AUC, this does not indicate the quality of our relevant snippets. We thus conduct a qualitative evaluation of snippets generated with an anchor-and-learn approach against an advanced keyword approach below in Table 6.6. The case study exposes a more fundamental limitation of the anchor-and-learn model– we do not incorporate any measure of recency into our model. Older, outdated snippets can be ranked higher than more recent ones if they contain highly predictive tokens, which can give a dangerously inaccurate view of the patient's disease history. Even if snippets generated using our anchor-and-learn relevancy

Concepts	Coefficients
hypertension	portal, hyperlipidemia, hld, hypercholesterolemia, hl, dyslipidemia, pulmonary, pmh, dm, high_cholesterol, hctz, pmhx, nci, diabetes, controlled, amlodipine, hydrochlorothiazide, atenolol, lisinopril, obesity
anxiety	depression, klonopin, panic_attacks, buspar, nos, panic, generalized, seroquel, situational, buspirone, valium, tremor, citalopram, psychotherapy
asthma	nebulizer, bronchitis, flare, pmhx, ventolin, prednisone, spirometry, action, pfts
renal transplant	kidney_disease, quadrant, end_stage_renal_disease, iga_nephropathy, iddm, pancreas, renal_failure, graft, blindness
c-section	pmhx, ob, labor, pregnancy, pshx, fetal, cholecystectomy, surgical, child

Table 6.5: Top-weighted Coefficients for Anchor-and-Learn Model for OMR Snippetization.

ranker do seem sensible at a surface level, we do not strike a practical balance between chronology and relevancy. We can attempt to remedy this via a number of ways, such as discretizing the relevancy of snippets and sorting snippets by date within these discrete chunks, or by systematically downweighting older snippets, but this is difficult to accomplish in a data-driven manner. Given this limitation, as well as inherent problems with disease comorbidities, we do not opt for this approach.

We note that anchor-and-learn logistic regression is simply one way to formalize binary snippet relevance– many other techniques exist for text classification, such as performing logistic regression on BERT-like contextual embeddings of the text [30], or using a nonlinear model for the learned classifier [69, 40]. However, these more complex methods lack clear interpretability to the physician, and as such, we did not consider them seriously for our initial pilot.

Standard Keyword Search	Anchor-and-Learn Logistic Regression
Condition: `Hyperlipidemia`	
October 18, 2011: Past history of a breast reduction, tummy tuck, bladder prolapse surgery. Right hand arthritis/tendinitis in wrist and along base of thumb. Borderline fatty liver hyperlipidemia (not well-controlled with diet or exercise). exercise and weight loss with her.	January 1, 2010: Mild hyperlipidemia with elevated cholesterol, but will recheck labs today. Reviewed diet
January 17, 2011: Osteopenia in spine. Also has borderline fatty liver, hyperlipidemia– diet is moderate in fat and cholesterol is high compared to baseline. Check lipids and LFTs today.	February 2, 2010: 51 y/o F with chronic medical issues: breast mass, hx +HPV on previous pap, no menses 4 years, mild hyperlipidemia, high cholesterol. Low Vitamin D on recent lab.
February 2, 2010: 51 y/o F with chronic medical issues: breast mass, hx +HPV on previous pap, no menses 4 years, mild hyperlipidemia, high cholesterol. Low Vitamin D on recent lab.	January 17, 2011: Osteopenia in spine. Also has borderline fatty liver, hyperlipidemia– diet is moderate in fat and cholesterol is high compared to baseline. Check lipids and LFTs today.

Table 6.6: Case Studies: Snippets surfaced with advanced keyword search vs. anchor-and-learn logistic regression for a sample condition.

6.4 Deployment and Next Steps

In order to efficiently implement our advanced keyword search and service requests quickly, we must carefully consider our design choices. We now discuss how we deploy our keyword search algorithm and room for future work.

6.4.1 Implementation of Advanced Keyword Search

Servicing Snippet Requests: Advanced keyword search can be implemented in a naive manner that does not prioritize efficiency.

Consider a patient whose OMR is split into an chronologically-ordered list S of snippets using the algorithm outlined in Section 6.2. We assume each snippet contains at most n words, each of which has at most m characters – then, for each snippet $s \in S$ we precompute the set K_s of keywords present in the snippet in total time $O(nm|S|)$ using a trie-based extraction algorithm. We maintain K_s as a hash table to allow for constant-time lookup.

Snippet requests are made automatically as a user enters text into the interface. From this text, a set C of concepts is extracted. We maintain a hash map from each concept c to a set K_c of keywords relevant to c, allowing us to compute K_c, which we maintain as a hash table, for each $c \in C$ in $O(|C|)$ time. To find snippets of the patient's medical history relevant to the concepts in C, we iterate through each $c \in C$ and each $s \in S$. For each c, s pair we determine that snippet s is relevant to concept c if the intersection $K_c \cap K_s$ is nonempty – this can be checked by searching for each element in K_c within K_s, or vice versa.

The runtime of this algorithm will depend on the sizes of the sets K_c and K_s, so we bound $\max_c |K_c| \leq N_c$ and $\max_s |K_s| \leq N_s$, and arrive at a runtime of $O(|C||S| \min(N_s, N_c))$, noting that we will only need to iterate over the smaller of K_c and K_s for each c, s pair.

In practice, this system leads to latency for users. Indeed, we find that re-iterating

through all snippets to service a request is sub-optimal. To ameliorate this issue and speed up the algorithm, we reformulate advanced keyword search as a *sparse matrix multiplication* problem, which allows us to find all snippets relevant to all conditions in a given request using an aggregated computation.

We reformulate the sets of keywords K_s in each snippet $s \in S$ as a binary matrix $A^S \in \{0,1\}^{|S| \times \kappa}$, where κ represents the total number of keywords in our clinical vocabulary. We set $A^S_{sk} = 1$ if $k \in K_s$ and 0 otherwise. We note that this matrix will be extremely sparse in practice, and therefore can efficiently store it in compressed sparse column format.

To find which snippets are relevant to a single concept c, we construct the sparse vector $v^c \in \{0,1\}^{\kappa}$, then set $v^c_k = 1$ if $k \in K_c$ and 0 otherwise. The product $A^S v^c$ will then be a vector of length $|S|$ whose nonzero entries correspond to relevant snippets. To extend this logic to a set of search concepts C, we construct a matrix $V^C \in \{0,1\}^{\kappa \times |C|}$ whose rows correspond to vectors v^c for each $c \in C$, and compute a matrix of snippet-concept relevance by taking the sparse matrix product $A^S V^C$ and examining its nonzero elements. Noting that $A^S V^C$ will have dimension $|S| \times |C|$, a nonzero element at indices s, c indicates that snippet s is relevant to concept c.

The runtime of the sparse matrix multiplication can be bounded as follows: for each of the $|C|$ columns of V^C, we must iterate through the at most N_s columns of A^S which contain a nonzero term in the position corresponding to that column. Then, we must sum up the at most N_s terms resulting from the product of each column of V^C with A^S. This implies a total runtime of $O(|C|N_s N_c)$ – in practice, we search across many snippets (i.e. $|S|$ is large) and the sizes of the sets K_c and K_s are quite small, and thus the sparse-matrix multiplication is significantly faster than the naive alternative.

This algorithmic speedup makes it possible to surface relevant snippets with minimal delay. In our implementation of this algorithm, we lazily compute A^S for each patient's OMR as we load snippets S into memory, and then compute and store the nonzero elements of the product $A^S V^C$, where the set C is defined to be the set of all

concepts in our condition ontology. Thus, we can lazily compute all snippet-concept relevance terms in time $O(|C|N_sN_c)$, and having done so can surface a list of snippets relevant to a given concept immediately by accessing the cached results.

Rendering Snippets in the UI: It is very unlikely that a user wants to or has the time to read through all snippets related to a particular condition, especially in the case of denser OMRs. It is thus unnecessary to send the full text for all relevant snippets at once. We instead send the minimum amount of information to identify each snippet– the index of the note it originated from, and its integer character spans. As a result, we respond to snippet requests with a list of integer tuples– `(note_index, char_start_index, char_end_index)`. We provide a separate API endpoint that returns the full text of a note given a note index. This allows us to virtualize our rendering of snippets in the UI– upon receiving a list of relevant snippets, the UI requests the full-length note text for the first few notes. The remaining notes (and their relevant snippets) are then only requested and displayed if/when the user explicitly asks to see them. This decreases the size of request/response payloads between the server and the client, and increases UI responsivity because we render exactly the data that is being used and no more. On average, across a variety of patients, the wall time of snippet requesting and rendering is around 3-5 seconds.

6.4.2 Future Work

The best way of knowing what snippets are relevant in a medical context is to get a direct reading of this, which we can do in our tool. By examining user metrics related to what snippets are explored further, we can ascertain exactly what notes, and more specifically note snippets, are important to the doctor. As an intermediate step, we can also use click-through audit data already available at BIDMC. This data is at a note- rather than snippet-level, but gives a high-level sense of the order of notes that a doctor clicks on. This would give us a better gauge of the relevancy/recency tradeoff as well as general note types that are helpful to the clinician.

Finally, we acknowledge that we have studied relevant OMR snippets for a particular condition, but doctors often browse through the OMR without a particular query in mind. It may be helpful to show OMR snippets from certain note types, subspecialties, or even section– as an example, we might surface all History of Present Illness sections for a doctor to get a quick overview of a patient's disease history.

Chapter 7

Information Synthesis and Visualization of Semi-Structured Data

In this chapter, we describe the contextual information retrieval and visualization utilities that we developed for the tool.

7.1 Drug-Disease Indications

While a doctor can ascertain all of the medications a patient is currently prescribed from structured data, it is difficult to automatically map each drug to the underlying condition it is treating. First, the doctor must aggregate a list of conditions that the patient has. Then, they use medical expertise to match drugs with conditions– this domain knowledge is a set of *drug-disease indications*. As an example, `metformin` treats `diabetes`.

Drug-disease indication datasets do exist, primarily because doctors cannot remember every drug on the market and must occasionally consult these datasets to confirm a mapping. However, they are stored in disparate sources that often cannot be easily

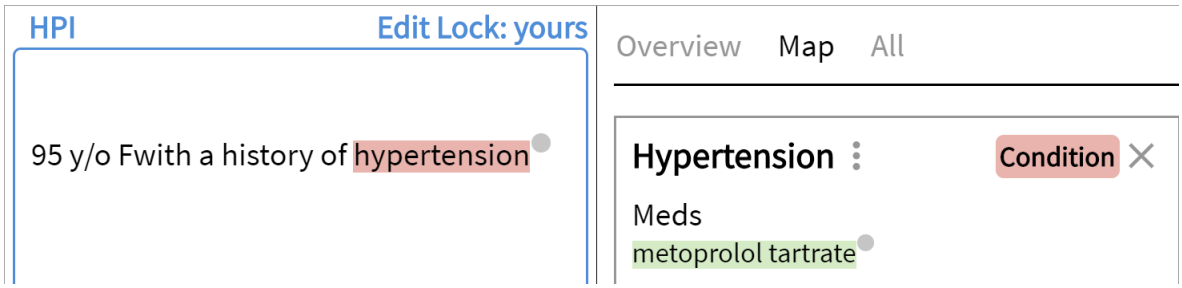


Figure 7-1: Screenshot of drug-disease indications in the system.

joined. The National Drug Formulary Reference Terminology (NDF-RT) database, for example, is mapped to UMLS CUIs, but FirstDataBank's DrugBank ontology uses a different identifier. Many other such datasets exist, including the Chemicals of Biological Interest from the European Bioinformatics Institute, the World Health Organization's Drug Dictionary, and the Prescriber's Digital Reference.

We use an aggregation of all of these sources as curated in [108]. This list of drug-disease indications is normalized to UMLS and contains over 192,000 unique drug-indication pairs for 30,000+ concepts. These indications are used in the tool to automatically match prescribed medications to tagged conditions without the doctor having to search through the medical record– in Figure 7-1, tagging `hypertension`, for example, surfaces `metoprolol tartrate` as a relevant medication. Clicking on `metoprolol tartrate` can then enable other contextual information retrieval such as dosage information and OMR notes with prescriber information.

7.2 Lab and Vital Trend Visualization

Clinicians often want to visualize quantitative lab and vital trends to establish a patient's baseline, compare it to reference values, and use this as a diagnostic tool. While historical lab and vital data is available, BIDMC's current EHR does not provide any in-build data aggregation or visualization.

We provide the user with multiple views of the data by relying on time-varying aggregation windows. Clinicians can visualize trends across the current visit, the last six

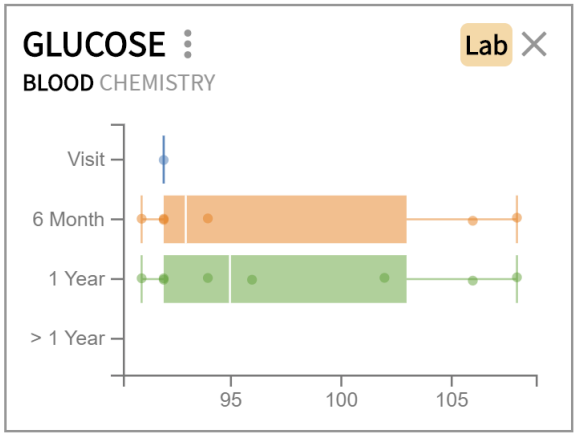


Figure 7-2: Box-and-whisker visualization of a lab trend.

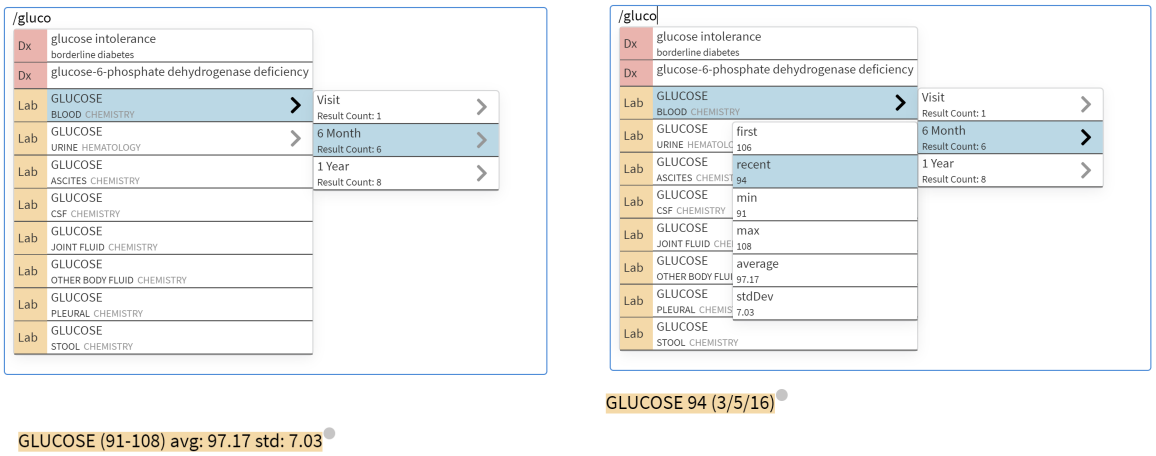
months, the last one year, and all available data. Tagging a lab or vital in the note will automatically add a box-and-whisker plot of the data to the sidebar as shown in Figure 7-2. The graph marks quartiles of the data distribution, as well as the individual lab/vital recordings for a given window. Comparing box-and-whisker plots over varying time windows gives a longitudinal view of how the trend has shifted.

In the low-data regime where a user has fewer than five vital/lab readings across all time, we instead opt for a standard tabular representation of the data instead of a graphical visualization, similar to what is proposed in [8].

When clinicians are mentioning a lab/vital trend in the note itself, we provide ways of aggregating lab trends over time into concise plaintext representations. Users can record the maximum, minimum, and mean lab/vital value (as well as standard deviation to get a measure of variance) using a multi-tiered autocomplete dropdown. An example of this is captured in Figure 7-3.

7.3 Condition-Procedure Relations

Medical conditions can be treated with drugs. However, chronic conditions often require surgeries or recurring treatments. If a patient presents with chest pain, any prior cardiac surgery is useful to know about. Similarly, if a patient has chronic



(a) Aggregate trends across time (b) Drill down into individual measures

Figure 7-3: Inserting aggregate lab/vital trends into the editor with autocomplete. (a) The user can use the first level of the autocomplete to select a timeframe to aggregate over. Pressing tab inserts text that encapsulates the range of values, the average, and the standard deviation. (b) The user can drill-down further to record individual measurements or summary statistics within a timeframe. In this screenshot, the most recent lab value within the last six months is recorded, along with its date.

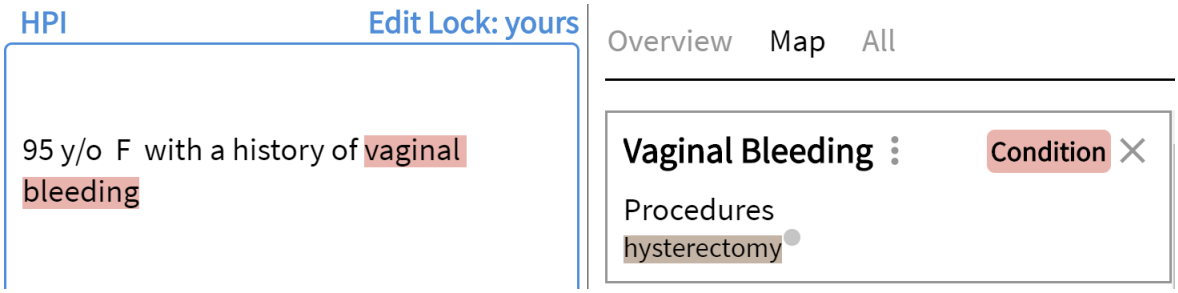


Figure 7-4: Screenshot of condition-procedure relations in the system.

kidney disease, clinicians care about whether or not they are on dialysis.

We thus wish to surface relevant procedures that are related to a condition in a structured manner. We define a *procedure* as a clinical concept that refers to a recurring treatment (chemotherapy, renal dialysis, not a drug), or a surgery (a distinct medical intervention that likely required an operation, e.g. liver transplant). An example of this in action is depicted in Figure 7-4, where we show that a patient has had a hysterectomy when the doctor mentions that the patient has a history of vaginal bleeding.

We first outline how to construct an ontology of procedures, and then describe various methods of learning a relationship between procedures and conditions. We ultimately adopt a machine translation approach in our live system, which converts representations of conditions into procedures and looks for neighbors in a latent space.

7.3.1 Constructing an Ontology of Procedures

To construct our ontology of procedures, we bootstrap from our condition ontology and fill in missingness by using UMLS concept types.

There are already many procedures embedded into our ontology of diseases/conditions, e.g. `cholecystectomy`, `tonsillectomy`, `chemotherapy`. We can find procedures in our condition ontology by checking whether its UMLS concept type includes "Therapeutic or Preventative Procedure." We then add all of these concepts to our procedure ontology unless:

- The term is too general (`immobilization`) or was incorrectly classified as a procedure by UMLS (`diabetic on diet only`).
- The term is too specific/rare and occurs fewer than 50 times across ≈270,000 of the ED notes we have in our retrospective dataset of clinical notes.

We then find any remaining procedures by running a trie-based capture of UMLS CUIs that met the following characteristics: (1) it was designated as a "Therapeutic or Preventative Procedure" UMLS concept; (2) it appeared as a procedure in SNOMED CT, which filters out nonsensical UMLS synonyms such as `water` for `hydrotherapy`; (3) its UMLS CUI appeared at least 50 times in our dataset of clinical notes. Finally, we manually group equivalent concepts and removed extraneous synonyms to tailor the ontology for the ED. As an example, `IVF` appears often in our dataset to represent `IV fluids`, whereas UMLS lists it as a synonym for `in-vitro fertilization`. Similarly, `left hip replacement` and `right hip replacement` are distinct UMLS CUIs but are condensed into a single concept in our ontology because they are

relevant in the same contexts. The final ontology consists of 112 concepts from 144 distinct UMLS CUIs. This represents 406 unique synonyms for procedural concepts.

7.3.2 Establishing Prior Mentions of Procedures

The simplest way of determining past procedures for a patient is to look for a string match of any procedure in the patient's OMR. However, this is a poor way of establishing a ground-truth because of hedging and uncertainty. Many notes will recommend surgeries and treatments, e.g. "the patient will likely need X surgery." The notes in our retrospective slice of data often do not have note titles or templated formats, so it can be difficult to find Procedural Notes (e.g. post-op surgery).

One way of detecting certain mentions of prior procedures is to use a rule-based approach with the following two heuristics.

1. If a term occurs in a sentence of the form `X: ...` for `X` in `[PMH, PMHx, HPI, hx of, PSH, Surgical History, Past Surgeries]`, it is certain.

2. If a term is immediately preceded by a local keyword such as `s/p, status post, status/post, complications with, underwent, had, previous since` (allowing for modifiers such as `right, left, partial` to be attached to the term), it is certain.

This works well in practice, especially given that even if a patient has one narrative note where a procedure is missed with our heuristics, it is highly likely that the procedure is mentioned in another note which we do capture. Procedures are mentioned in approximately 15% of ED notes in our system, and 56% of these mentions are classified as certain using this algorithm. Over 50% of ED notes with a certain procedure mentioned had at least one certain mention of the procedure in a prior OMR note.

To learn this more generally, we might pose the problem differently: consider some medical context (triage text, ED note) C. For a particular procedure P, we find all

sentences S in the OMR where P is mentioned. Then, we learn a mapping from $(S, C) \rightarrow \{0, 1\}$ to determine whether the patient truly had the procedure. We can generate positive labels for the task if P was mentioned in a corresponding ED note, and with our rules. However, generating negative labels is more nuanced because presence of P within an ED note does not represent the ground truth. In many cases, doctors choose to omit medically-relevant information because they are pressed for time, or because they neglect to find it in the OMR. In addition, the dataset of ED notes we have represents early draft versions of clinical notes, potentially before relevant prior procedures are added. Given that in future iterations of the tool we might receive structured procedure data in the form of Current Procedural Terminology (CPT) codes, we find that the rule-based approach works well as a stopgap to extract procedures from a patient's OMR.

7.3.3 Mapping Conditions to Procedures via Semi-Supervised Affine Transformations

A core problem we face is that while a mention of a procedure in an ED note indicates that it was relevant, it is difficult to ascertain which condition(s) made it so. In addition, the absence of a procedure in our dataset doesn't indicate that it was irrelevant to the clinical context– it's possible that the doctor wanted to write a concise note, or did not notice the procedure had happened.

However, procedures usually relate to a condition usually when the procedure affects a body part or organ system that the condition affects. Prior cardiac surgery, for example, is automatically relevant for any condition related to the heart.

One way to incorporate medical knowledge in our model while simultaneously ignoring the bias of procedures mentioned in ED notes is to use embeddings of concepts trained on external corpora. In a simple baseline, we can take a condition concept c and look for its nearest neighbors among procedures in some latent space. We use embeddings from [13], who train Word2Vec-style embeddings for UMLS CUIs using a selection of

medical claims data, clinical notes, and biomedical journal articles.

Note that the naive approach of looking for nearest neighbors to a concept does not exploit any of the labeled data we have. For example, we know that if a procedure was mentioned in an ED note as well as in a patient's OMR, the procedure was specifically relevant within the context of other clinical concepts mentioned in the ED note. That said, if a procedure is not mentioned in the ED note, it is only a weak indication that it isn't relevant. Essentially, mentions of procedures in ED notes are an incomplete subset of the ground-truth relationships between conditions and procedures.

Inspired by older work in semi-supervised machine translation such as [4], we formulate condition-procedure relations as a noisy machine translation task. We utilize concurrent mentions of procedures and conditions in ED notes to learn an affine transformation from conditions in an embedded space to procedures in an embedded space. Then, we find relevant procedures to a particular condition by transforming the condition into our procedure space and looking for its nearest neighbors.

Formally, consider an clinical note with a set of documented conditions C and procedures P. We define K affine transformations of the form $f_i : v \rightarrow A_i v + b_i$ for $i \in \{1, \cdots, K\}$. We first transform each of our conditions $c \in C$ into our procedure space with each of our K transformations, creating a new set of points $C' = \cup_{i \in \{0, \cdots, K\}} \{f_i(c) | c \in C\}$. The translated conditions should be close to related procedures P in our embedded space, so we define our loss function as

$$\ell = \sum_{p \in P} \min_{c' \in C'} d(p, c')$$

for some appropriately chosen distance function d. We minimize this objective using stochastic gradient descent to learn A_i and b_i for each of the K transformations, and use Euclidean distance for d. Defining multiple transformations allows us to have different modes of converting conditions to procedures.

We also examine results when forcing each $A_i = \mathbf{I}$ (i.e. converting conditions solely by adding a bias vector) as well as adding a regularization term $||A_i - \mathbf{I}||$ into ℓ to prevent

each A_i from deviating from the identity, using a Frobenius norm. In practice, we only use procedures that are mentioned in the ED as well as in the OMR to remove any uncertain mentions of procedures.

We compare relationships learned from this method against the naive nearest-neighbors approach below in Figure 7.1, and find that the affine transformations correct for issues in the embeddings provided by [13] and give medically-meaningful results. Note that training the affine transformations solely on mentions of concepts within OMR notes (and not including ED notes) does not make a qualitative difference in our model, which we tested. In practice, we deploy the learned affine transformation method setting $K = 2$, and only display a procedure as relevant to a condition if (1) the patient had a certain mention of the procedure in the OMR, and (2) that the procedure is within the top ten closest neighbors to the transformed condition.

7.3.4 Mapping Conditions to Procedures via Logistic Regression

Instead of creating a ranking over procedures that lie close to a condition in some embedded space, we might attempt to learn the binary relevance of a procedure p to a set of conditions C directly, e.g. a function $f : (C, p) \to \{0, 1\}$. We can generate training data for this task as follows: For patient i, we detect all past procedures P_i a patient has had by looking for certain mentions of procedures in prior OMR notes (using our rule-based approach to detect certainty). We also find all conditions C_i mentioned in a corresponding ED note. Then for each $p_i \in P_i$, we featurize (C_i, p_i) in some way and attempt to predict whether p_i appears in a corresponding ED note. If p_i does appear, it is clearly a positive label as the procedure was relevant in the given medical context. However, if it does not appear, we can either treat it as a negative label or as an unknown label.

Note that $|C_i|$, or the number of concepts present in an ED note, can vary. In order to featurize (C_i, p_i) to feed into a model like logistic regression, we must featurize C_i with

Naive NN	Learned Bias	Learned Affine
	Diabetes mellitus	
cadaveric renal transplant	dialysis procedure	coronary artery bypass
antiretroviral therapy	subtotal pancreatectomy	cardiac surgery procedures
dialysis procedure	bilateral total nephrectomy	perc. coronary intervention
pancreas transplantation	pancreas transplantation	coronary angioplasty
parathyroidectomy	gastric bypass	repair of aneurysm
cervical arthrodesis	cadaveric renal transplant	dialysis procedure
bilateral mastectomy	peritoneal dialysis	valvuloplasty of aorta
subtotal pancreatectomy	roux-en-y gastrojejunostomy	pancreas transplantation
	Deep Vein Thrombosis (DVT)	
carotid endarterectomy	thrombectomy	repair of aneurysm
thrombectomy	repair of aneurysm	thrombectomy
percut coronary intervention	carotid endarterectomy	carotid endarterectomy
pancreas transplantation	bypass graft	bypass graft
coronary angioplasty	placement of stent	angioplasty
cadaveric renal transplant	angioplasty	transplantation of liver
dialysis procedure	perc. coronary intervention	perc coronary intervention
bypass graft	coronary angioplasty	placement of stent
transplantation of liver	blood transfusion	dialysis procedure
	Gastritis	
subtotal pancreatectomy	roux-en-y gastrojejunostomy	total colectomy
distal pancreatectomy	proctocolectomy	proctocolectomy
roux-en-y gastrojejunostomy	total colectomy	roux-en-y gastrojejunostomy
proctocolectomy	resection of polyp	resection of polyp
pancreas transplantation	subtotal pancreatectomy	subtotal pancreatectomy
total colectomy	distal pancreatectomy	distal pancreatectomy
loop ileostomy	loop ileostomy	small intestine excision
cholecystostomy	pancreatectomy	pancreatectomy

Table 7.1: Qualitative examples of closest procedures to conditions using affine transformations. The naive nearest-neighbors baseline looks for nearest procedures to the given condition using Euclidean distance between word2vec embeddings. The Learned Additive Bias and Learned Affine Transformation methods learn K bias vectors or a linear transformations to transform conditions respectively, and then finds nearest procedures to the transformed condition using Euclidean distance between word2vec embeddings. We show results for $K = 2$.

a fixed-length vector that is agnostic to $|C_i|$. We experiment with different pooling strategies, including mean- and max- pooling concept vectors for each concept in C_i. We then concatenate this pooled vector with an embedding of p_i and feed this as input into our classifier. We experiment with treating lack of mentions of a procedure in a note as both a negative label and unknown label. In the latter paradigm, we use the framework of [35], which shows that the conditional probabilities produced by a model trained on both labeled and unlabeled examples differ by only a constant factor from the conditional probabilities produced by a model trained on fully labeled positive and negative samples.

However, using these methods, we find that classifiers trained using this methodology have poor qualitative results, especially compared with our semi-supervised affine transformation approach, and thus did not pursue this avenue.

7.4 Differential Diagnosis

The ambitious vision of how machine learning can transform EHRs outlined in [70] identifies differential diagnosis as a core part of clinical decision support. *Differential diagnosis* is the process of proposing and ruling out different conditions that cause the patient's symptoms. Expert diagnostic support systems have been extensively researched, but are unsurprisingly hard to extend [10]. Learned models, on the other hand, can encode complex patterns from large datasets and have gained traction in recent years. However, these models often struggle to incorporate expert advice [96, 12, 95] or generalize beyond specific chief complaints or disease types [73].

Structured data that is created as the doctor types a note can be used to build better differential diagnosis models. While we do not currently support differential diagnosis in the tool, we outline initial prototypes of two diagnostic models: a general model designed to predict a patient's primary ICD code; and a chief complaint-specific model that predicts a clinically-derived diagnosis given a chief complaint of abdominal pain. The former ideally aids in general brainstorming for possible diagnoses in case a

clinician misses an important clue, while the latter helps narrow and prioritize the correct diagnosis given a specific use case. We also build a tool that allows a doctor to visualize these predictions as they compose a note as a preliminary demonstration of the utility of differential diagnosis support.

For this initial analysis, we focus on *discriminative* approaches to predicting diagnoses, rather than building a generative causal model to map underlying conditions to present and absent symptoms. This allows us to rapidly iterate on models. Ultimately, we see three potential mechanisms to integrate machine learned differential diagnosis in the tool: (1) predicting likely diseases given a set of presenting symptoms, and then finding snippets related to those diseases in the sidebar; (2) updating the autocomplete based on already documented symptoms; and (3) prepopulating the `Diagnosis` section of the note with suggestions.

7.4.1 Generalized Differential Diagnosis

We first seek to develop a single unified model for differential diagnosis. This model operates at the level of ICD billing codes, and has the objective of predicting what the primary diagnosis code will be using information collected in the EHR. Due to its universality, this model obviates the need for excessive manual data processing, and allows us to immediately provide meaningful suggestions for the overwhelming majority of patients.

Creating a Dataset

In a patient's discharge summary, there are two clues into the final diagnosis. The first is the structured International Classification of Diseases-9 (ICD9) billing codes which are assigned per visit. All of the visits in our dataset are assigned an ordered sequence of codes, but not all codes are relevant to or representative of the underlying condition– as an example, a diabetic patient who enters the ED complaining of arm pain may be billed for diabetes if their treatment related to the disease, even if it

was not the original reason for coming in. ICD codes are also assigned to each visit by a panel of medical billing experts; these codes do not necessarily align with the diagnostic mental model that doctors use during the course of treatment. However, because these codes are structured, we can leverage already-existing ontologies and hierarchies in our models.

The second, more direct clue into the diagnosis is the `Diagnosis:` section of the clinical note, which often contains a few phrases that encapsulate the diagnosis. Clinical notes in our retrospective dataset do not always contain this section, though. After extracting any symptom/condition concepts from our ontology that appeared following mentions of `Dx:` or `Diagnosis:` from raw ED notes, only 68 concepts were mentioned more than 100 times. In contrast, there are 373 ICD9 codes that appear more than 100 times in our dataset.

Due to difficulties in extracting diagnoses from the raw ED note text itself, we use *primary ICD9 codes* as our labels. ICD codes are ordered per visit, and while the secondary codes usually correspond to chronic conditions such as diabetes or hypertension that are crucial to the given visit, the first, primary code is usually representative of the true diagnosis. While the tail of ICD9 codes is long (there are around 3500 ICD codes in total in the dataset), these 373 labels represent 87% of the distribution of primary codes.

We seek to aid the doctor by providing differential diagnosis support at each stage of medical decision making. Our model should provide helpful diagnostic suggestions based on the triage note even when the doctor has not yet seen the patient, as well as certainty about the correct diagnosis once it is documented. We simulate this by creating an augmented dataset comprised of the following types of text input:

- Entire notes (triage notes concatenated with `MDcomments`): the use case here is that a clinician is largely finished documenting their findings, and wishes to enter a final diagnosis.

- Triage notes alone: this allows us to learn how to predict a (possibly vague)

diagnosis based solely on preliminary findings. Qualitatively, the triage note often contains enough information to make a good diagnosis in simpler cases, and we want to be able to capture that in our model.

- Triage notes concatenated with a random prefix of the clinical note, chosen uniformly between 10 words into the note to full length. This type of training data lets us account for the use case of a doctor wanting diagnostic suggestions while in the midst of writing a note.

A future direction of research might be to use audit data to find reasonable "checkpoints" in the note to predict from, rather than random subsamples.

Modeling & Results

We experiment with a mix of linear models and nonlinear neural networks to perform our diagnostic prediction.

Linear Models: We first train logistic regression models to make a multiclass prediction over our each of our primary ICD code classes, using various tokenizations and featurizations of the text:

1. A binary vector of all symptoms and conditions indicating whether they were positively mentioned in the note (negative mentions are treated the same as an absence of a mention).

2. A multihot representation of each symptom and condition indicating whether it was positively mentioned, negatively mentioned, or absent. Each concept is essentially treated as a ternary variable.

3. A TF-IDF vectorization of a bag-of-words representation of the text. Within this, we explore two tokenization strategies: "standard" tokens that are space-delimited, and "condensed" tokens where concepts and their negations are treated as single concepts (e.g. `chest_pain`, `no_chest_pain`).

Multi-Branch 1-D CNNs: We also explore convolutional neural networks to test how nonlinear models would fare. Intuitively, the success of a bag-of-words featurization for linear models hints at the fact that certain words or combinations of words are indicative of final diagnoses. One way to generalize this beyond bag-of-words is to train a one-dimensional CNN on an embeddings of the triage and clinical note texts. State-of-the-art for ICD code prediction from text use a 1-D CNN, augmented either with an attention layer for interpretability [83] or with residual network to capture longer-term dependencies [67].

Inspired by these architectures, we choose a one-dimensional *multi-branch CNN* model. First, we map each token in our vocabulary to a d-dimensional vector embedding. Then we, choose n_k filter sizes $k_1, k_2, \cdots, k_{n_k}$ (hyperparameter tuned). For each filter size, we create n_f filters of the given length such that filters corresponding to a length of k_i are represented by a $k_i \times d$ matrix of trainable real numbers. Each filter is convolved with a stacked embedding of each word in the combined triage and clinical note text. A max-pooling is used per filter of size k_i to pick the single highest cross-correlation between any contiguous k_i words in the filter matrix. These max-pooled outputs are then concatenate into a vector of length $n_f \times n_k$, which is then passed through a ReLU activation and a fully connected layer. This maps the input to a vector whose length is equal to the number of possible diagnoses (classes). A softmax is finally applied to get estimates of the probabilities of each class.

There are two modes in which this model can improve over the bag-of-words baselines. First, longer filters can learn multi-token phrases of relevance. Second, these filters can act as "soft" keywords, allowing for a broader group of ideas that coexist in the same region of the embedded space to share weights and be used in a similar manner for prediction, thereby improving generalization.

The model is trained end-to-end with a weighted cross-entropy loss due to class imbalance. Embeddings are initially seeded using a word2vec model trained on OMR notes, but are trained in tandem with the CNN. A full list of tunable hyperparameters that we searched over when training models are shown in Table A.1. Because this ar-

Model	MRR ↑	Accuracy ↑
Rank by frequency (deterministic)	0.14	1%
Logistic Regression (binary representation of symptoms/conditions)	0.35	19%
Logistic Regression (ternary representation of symptoms/conditions)	0.33	21%
Logistic Regression (BoW of text alone, standard tokenization)	0.43	28%
Logistic Regression (BoW of text alone, condensed tokenization)	0.48	31%
Multi-branch CNN (word2vec embeddings of each token in text; 600 length-1 filters)	0.52	36%

Table 7.2: Performance of diagnostic models for general differential diagnosis. Measured by mean reciprocal rank (MRR) and accuracy across the entire dataset, which contains each triage note as well as a random subsample of the clinical note.

chitecture requires a fixed-length input, we truncate input text to a constant number of tokens and pad the text with whitespace if necessary. We find that an input length of 512 tokens achieves the best performance for our generalized differential diagnosis model, and truncates only 0.12% of the dataset.

Results: We use two performance metrics to evaluate our model: the *mean reciprocal rank* (MRR) as defined in Section 4.4.1, and overall accuracy. The MRR allows us to measure the quality of our rankings from an informational retrieval perspective, while accuracy is a harsher gauge of our model performance. Results as measured on a held-out test set of visits are shown in Table 7.2. Sampling random prefixes of the `MDcomments` to simulate diagnosis over the course of a visit also helps performance compared to only using full note texts. This is particularly apparent in the best performing multi-branch CNN, as shown in Figure 7-5. This model, which learns 600 length-one filters, is effectively a learned keyword search. The input tokens into the model already capture multi-word clinical concepts and negated terms, which is likely why the model did not need to learn longer filters. To more precisely establish what words this model is capturing, we measure the cross-correlation between all of our word embeddings and each filter matrix to find salient patterns. Note that we define

Correlated Tokens per Filter			
hearing_voices	headache	neb	no_fevers
telling	ha	nasal	no_fever
disorganized	myalgias	benadryl	no_fc
paranoid	migraine	combivent	no_fevers/chills
manic	photophobia	epi	f/c/s
auditory_hallucinations	frontal_headache	nebs	no_fever/chills
paranoia	sore_throat	solumedrol	septic
delusional	neck_stiffness	nebulizer	fevers
sane	sinusitis	hives	pyelo
disorder	feel	albuterol	no_abscess

Table 7.3: Highly correlated tokens for a selection of filters in the generalized differential diagnosis CNN model.

the cross-correlation for two real-valued, length-k vectors $\mathbf{u}, \mathbf{v}$ as

$$\text{xcorr}(\mathbf{u}, \mathbf{v}) = \sum_{m=0}^{k} \mathbf{u}[m]\mathbf{v}[m-k]$$

Many filters clearly encapsulate closely-related medical concepts, and we can delve further into what drives model predictions by examining highly correlated word embeddings for each filter, as listed in Table 7.3.

While the MRR of this model is relatively high, indicating that the correct diagnosis is, on average, the first or second suggestion, our set of ICD codes is restrictive– we only predict from a set 373 relatively common codes. One way to mitigate this within our current framework is to build a hierarchical model and roll up rarer ICD codes to a coarser level of the coding hierarchy. Alternatively, given sufficient data, we could simply expand our set of labels to include infrequent codes.

In practice, we also obscure the fact that we are predicting at a code level. Our goal in building a new EHR is not to automate billing, but to provide diagnostic support to physicians– while we use ICD codes as a proxy for the true diagnosis, they do not always represent the ground-truth. Moreover, doctors do not diagnose patients in line with ICD codes. As such, we also manually curate a mapping from each ICD code to natural language, and display these human-readable names instead. As an example,

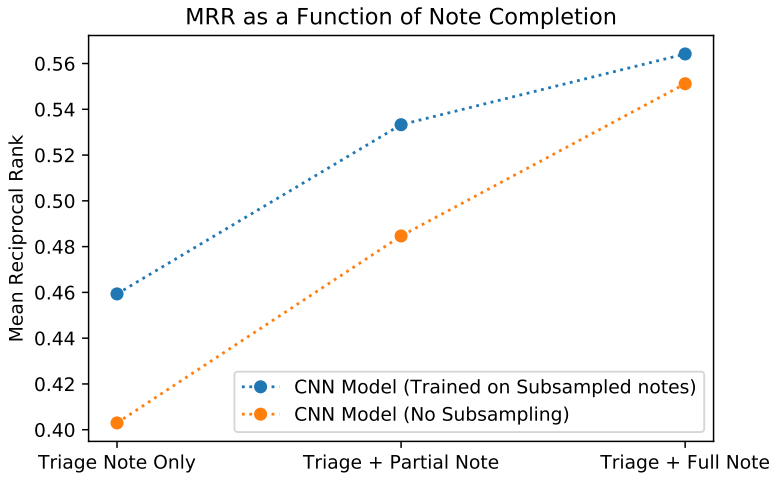


Figure 7-5: MRR as a function of note length for best-performing multi-branch CNN models trained on full length versus subsampled notes.

the ICD9 code `592.0 (CALCULUS OF KIDNEY)` is mapped to the more colloquial `kidney stone`.

Future work into this tool can use interpretability methods for CNN text classification to identify snippets of the note that prompt the model to make a given prediction, and incorporate structured data like lab tests. Separately, we might frame this as a continuous problem, where instead of predicting a single relevant ICD code, we predict a relevant point in a vector space of ICD codes and look for nearest neighbors.

7.4.2 Differential Diagnosis of Abdominal Pain

To complement our general differential diagnosis approach, we also develop models to predict clinically-driven diagnoses from a given chief complaint. This is more in line with classical differential diagnosis literature. In these expert-mediated processes, the chief complaint governs the sequence of rules and tests that are used to determine the final diagnosis [38, 115]. To mimic this set-up, we carefully derive diagnostic labels for visits with a particular chief complaint using a combination of ICD codes and EHR data, and predict these labels from the triage text and clinical note.

For the purposes of an initial prototype, we focus on a chief complaint of abdominal pain and its derivatives (`ruq pain, epigastric pain`), which is the case for around $25,000$ visits. Based on our dataset as well as literature on differential diagnosis of abdominal pain in the ED [81, 57], we settle on eighteen possible diagnoses.

Deriving Ground-Truth Diagnostic Labels

In order to aid in differential diagnoses, we must label each visit whose chief complaint was abdominal pain with one of our eighteen final diagnoses. We do so by making use of ICD codes.

First, we pre-process our potential diagnoses to express them in terms of ICD codes. We establish which ICD codes correspond to each of the eighteen diagnoses. Note that due to the high specificity of ICD codes, many codes often roll up into a single diagnosis. Next, we sort the potential diagnoses by specificity, which we are able to perform manually based on medical literature – as an example, we would rank the generic term "abdominal pain" after the very specific condition "diverticulitis". A natural next step here would be to learn this specificity order in a data-driven way; however, due to the low number of potential diagnoses and the easy availability of relevant medical literature, we opted for a manual approach.

We label patient visits by iterating through the codes associated with the visit, and checking if each code belongs to any of the groups of ICD codes corresponding to a diagnosis. In the case of multiple matches, we select the diagnosis which is the most specific. This reflects the clinical intuition that if a patient exhibits a precisely defined disease, then diagnosing them with this disease is more informative than using a more general term. As a second line to get more informative labels in cases where the diagnosis codes are incomplete or nonspecific, we look for explicit mentions of string templates corresponding to diagnosis, such as `Diagnosis: GERD` in the text – if no such mentions are found, we finally fall back to the generic diagnosis of "nonspecific abdominal pain". We show each of our diagnostic labels, the ICD codes associated with each one, and the class size in Table 7.4.

Diagnosis	ICD9 Codes	N
Uterolithiasis	592.0 (CALCULUS OF KIDNEY), 594.1 (BLADDER CALCULUS NEC), 592.1 (CALCULUS OF URETER)	1277
Pancreatitis	577.0-2 (ACUTE/CHRONIC PANCREATITIS)	1215
Urinary Tract Infections	590.80 (PYELONEPHRITIS NOS), 599.0 (URIN TRACT INFECTION NOS), 590.10 (AC PYELONEPHRITIS NOS)	845
Diverticulitis	562.00-11 (DIVERTICULITIS S INTEST NO HEM)	617
Appendicitis	541-542 (APPENDICITIS NOS), 540.9 (ACUTE APPENDICITIS NOS)	607
Cholangitis	576.1 (CHOLANGITIS)	100
Cholelithiasis	574.XX (CHOLELITHIASIS)	583
Cholecystitis	575.XX (CHOLECYSTITIS)	257
Gastroesophageal Reflux Disease (GERD)	530.11 (REFLUX ESOPHAGITIS), 530.10 (ESOPHAGITIS UNSPECIFIED), 530.12 (ACUTE ESOPHAGITIS), 530.81 (ESOPHAGEAL REFLUX)	68
Gastroparesis	536.3 (GASTROPARESIS)	121
Gastritis	535.30 (ALCOHOL GASTRITIS-NO HEMORRHAGE), 535.70 (EOSINOPHILIC GASTRITIS, WITHOUT MENTION OF HEMORRHAGE), 535.50 (GASTRITIS/GASTRODUODEN-NO HEM), 535.40 (GASTRITIS NEC-NO HEMORRHAGE)	101
Gastroenteritis	555.9 (REGIONAL ENTERITIS NOS), 789.06 (ABDOMINAL PAIN EPIGASTRIC), 558.9 (NONINF GASTROENTERIT NEC)	2211
Bowel obstructions	576.2 (OBSTRUCTION OF BILE DUCT), 552.1 (UMBILICAL HERNIA W OBSTR), 560.XX (INTESTINAL OBSTRUCT)	751
Ovarian cysts	617.XX (ENDOMETRIOSIS), 620.2 (OVARIAN CYST NEC/NOS)	253
Inflammatory/Irritable Bowel Diseases	556.5-9 (ULCERATIVE COLITIS), 564.1 (IRRITABLE COLON)	77
Gastrointestinal bleeds	578.9 (GASTROINTEST HEMORR NOS)	126
Female pelvic inflammatory diseases	614.0-9 (CHRONIC PELVIC INFLAMMATORY DISEASE), 131.01 (TRICHOMONAL VAGINITIS), 218.9 (UTERINE LEIOMYOMA NOS), 625.8 (FEM GENITAL SYMPTOMS NEC), 615.0 (AC UTERINE INFLAMMATION), 625.9 (FEM GENITAL SYMPTOMS NOS), 112.1 (CANDIDAL VULVOVAGINITIS), 616.10 (VAGINITIS NOS)	281
Unspecified abdominal pain	789.00-09 (ABDOMINAL PAIN NOS)	12864

Table 7.4: Diagnostic labels for chief complaint of abdominal pain or related. Labels are ordered by specificity, such that if a visit is assigned a ICD code from two or more rows, it is labeled with the more specific diagnosis. N represents the number of samples in the given class.

Modeling & Results

We seek the same goal of predicting an `abdominal pain`-specific diagnosis at various stages of medical decision making, and featurize our model with the triage text as well a randomly subsampled prefix of the clinical note. We develop a mix of linear and nonlinear models as before.

We first train logistic regression models to make a multiclass prediction over our eighteen classes, using the same tokenization and featurization strategies outlined in our generalized differential diagnosis tool above. We also add a fourth linear baseline: a TF-IDF vectorization of a bag-of-words representation of the text, concatenated with a binary vector of which concepts were mentioned in the patient's OMR. This is an initial exploration into how the medical record can be used to improve differential diagnosis by factoring in prior diseases.

We also conduct an architecture search over multi-branched one-dimensional CNNs as described in Section 7.4.1. These models implicitly train embeddings of each token that are seeded from word2vec embeddings trained on OMR notes. These are small but nonlinear models. As a deterministic baseline, we rank diagnoses by their frequency of occurrence.

We again measure performance using MRR and accuracy. Results are shown in Table 7.5. Performance of the best model (the multibranch CNN) as a function of note length is shown in Figure 7-6. We note that the high accuracy/MRR of the deterministic baseline is due to of the heavy class imbalance in our dataset, and posit that many cases where `unspecified abdominal pain` is listed as the only diagnosis rather than a more specific condition is likely a coding error– thus, our labelling strategy may not always be capturing the true underlying cause of the patient's presentation.

The CNN model that we develop also lends itself well to local introspection. While any logistic regression model can capture obvious relationships between the text and the diagnosis (e.g. `rlq pain` and `appy` indicating a diagnosis of appendicitis), the CNN

can capture *sequences* of tokens. Our convolutional architecture effectively featurizes sequences of text that match up with learned filters as real scalar values, which are then fed into a single dense layer for prediction. This means that we can immediately identify what CNN outputs are relevant to the prediction of a certain diagnosis by finding the outputs with highest weights in the final linear layer. Furthermore, since each of the CNN outputs is the result of a max-pooling operation, it can uniquely be identified as the consequence of the presence of a substring in the input text. As a demonstrative case study, consider the following note text for an ED visit:

```
TRIAGE: Pt ate pork for dinner. Developed pain about
3 hours ago.  +nausea, normal bowel movements.

CLINICAL NOTE: 30F with abdominal pain and nausea.
Nausea started one hour after eating pork chop dinner.
Abd pain started 3 hours ago. PTA to ED.
Multiple family members ate pork without similar symptoms.
Patient should not be eating spicy foods, but ate
spicy food tonight. Nausea but no vomiting.
BM nml consistency and without blood.
Epigastrium TTP generalized upper abdominal pain.
```

While the diagnosis is nonobvious from this information, our model correctly predicts `cholecystitis`. Using the above introspection technique, we identified the top phrases that drove this prediction: `one hour after eating`, `after eating pork chop`, `generalized upper abd pain`. This matches clinical evidence indicating that cholecystitis signs often occur after large or fatty meals.

Model	MRR ↑	Accuracy ↑
Rank by frequency (deterministic)	0.42	**56%**
Logistic Regression (binary representation of symptoms/conditions)	0.51	30%
Logistic Regression (ternary representation of symptoms/conditions)	0.44	17%
Logistic Regression (BoW of text alone)	0.63	38%
Logistic Regression (BoW of text, binary representation of conditions in OMR)	0.57	24%
Multibranch CNN (word2vec embeddings of each token in text; kernel sizes 1, 2, 3, 4; 100 kernels per branch)	**0.73**	**56%**

Table 7.5: Performance of diagnostic models for a chief complaint related to `abdominal pain`. Measured by mean reciprocal rank (MRR) and accuracy across the entire dataset, which contains each triage note as well as a random subsample of the clinical note.

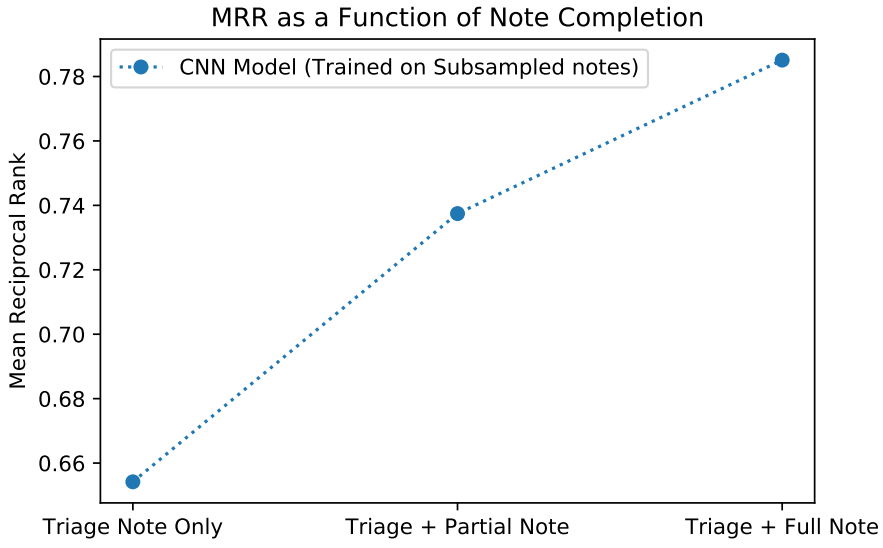


Figure 7-6: Performance of multibranch-CNN to predict diagnosis for a chief complaint of abdominal pain, broken down by note length.

7.5 Future Work

There are many other ways to provide clinical decision support for doctors that are not currently supported in our pilot version of the tool. Here, we briefly mention other avenues to improve clinical decision support within the tool.

- Instead of hard-coding important disease-lab and symptom-lab relationships, learn these directly.
- Add the ability to order sets of labs or treatments, directly from the tool. Notify the clinician when the order is ready.
- Use data collected with the tool regarding the utility of OMR snippets to train a fully-supervised text classifier to find relevant OMR snippets.
- Provide easy access to common educational resources that a doctor might want to use, including links to relevant topics on UpToDate. This is especially useful at BIDMC, which is a teaching hospital.
- Display a patient health bar with colors to indicate risk for common patient outcomes. There is a wealth of literature on risk stratification to borrow from here– both for specific conditions as well as for general outcomes like hospital admission and mortality [25, 88, 2].
- As a doctor is documenting the treatment plan for the patient, highlight parts of the medical history that might affect the outcome of the proposed treatments. This is a more involved version of OMR snippetization. This can also be abstracted to building patient treatment timelines– given a disease, what therapies has the patient tried?
- Inspired by Nigam Shah's "Green Button" metaphor [44], find similar patients to the one the doctor is currently treating, driven by a distance function that uses both structured and unstructured data in a patient's medical record. This can inform treatment guidelines and cohort construction.

- Meta-planning about hospital administration– using ward-level information about what beds each patient is assigned to, identify clusters of infectious patients that should be quarantined.

Chapter 8

Implementation & Deployment

8.1 System Implementation

The overall system, which is detailed in Figure 8-1, is comprised of three components:

1. *Client UI*: The frontend client is a React/Node.js app that houses the UI for the project. It can send and receive requests to/from the clinical server. The UI is rendered as an HTML iFrame within BIDMC's custom EHR interface. For security purposes, the client can only connect to the clinical server and does not directly communicate with the ML server. We use Facebook's Draft.js as our underlying editor framework, which is a rich text editor written for React.

2. *Clinical Server*: The clinical server is the only component with direct access to the BIDMC's clinical database. It can communicate both with the client UI but also with the ML server. In order for the frontend to communicate with the ML server, it sends indirect requests to the clinical server via a tunnel endpoint, which is then rerouted directly to the ML server. Responses are forwarded back to the client UI. Tunnel requests are one-way and can only be initiated from the client UI. The ML server has no ability to send requests and can only serve them.

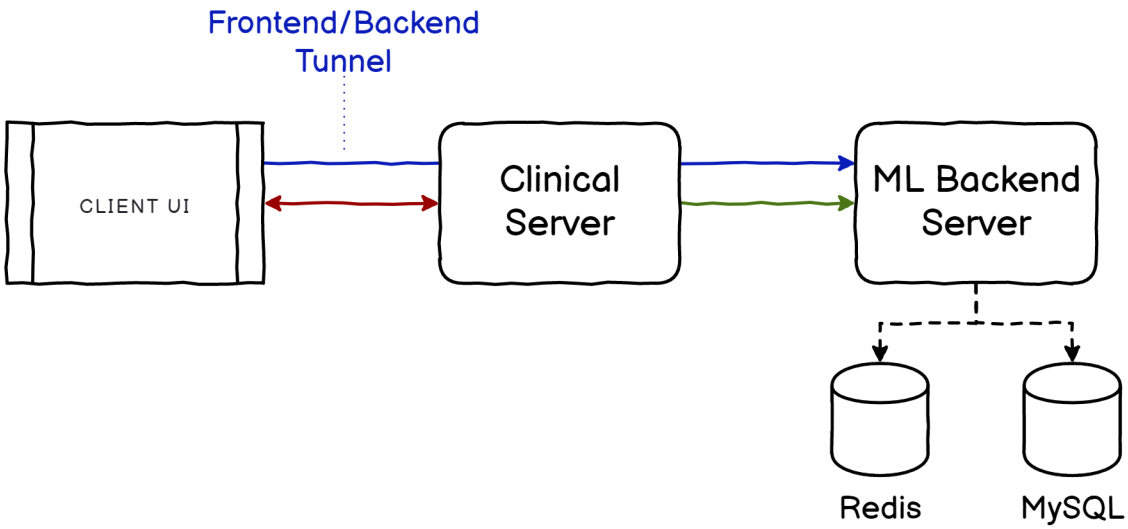


Figure 8-1: System Infrastructure

3. *ML Backend Server*: The backend server is a Dockerized Python/Flask app that houses the backend for the project. It sits on BIDMC's research server. All model weights, and any information derived from clinical data (e.g. population statistics) is stored here. It also supports both temporary and disk-persistent and temporary storage of data via Redis and MySQL respectively. It runs Python 2.7 on Ubuntu 16.04.

In order to persist and cache data for performance reasons, we have two additional storage mechanisms. We use Redis as an open-source, in-memory data structure store to cache data. The Redis store exists as a Docker container and can only be accessed via the Docker network running on the ML Backend Server– it keeps JSON serialized data that are keyed by strings, and caches patient information for use by the ML server, responses to frequent requests, and copies of data for the Client UI to access.

We also use MySQL to store persistently. The data within MySQL is saved to disk as a docker volume and only houses two tables: user metrics, and saved copies of notes. API documentation and internal datatypes and schemas used to send and store data are located here.

8.1.1 Client-Side Inference

In general, our backend server is used to persistently cache/store content and also to perform a Python environment to invoke machine learning models. In the case of autocomplete scope/type detection for contextual autocomplete, however, we must call our model as soon as a user appends words to the text. This forces us to run the model client-side in JavaScript.

On page load, the client sends a request to the ML server backend for the model configuration parameters and weights. The embeddings must be requested in batches because it is too large of a JSON payload to send at once. The client then loads model weights into a `Tensorflow.js`-based neural network, which takes approximately 5-6 seconds. We force `Tensorflow.js` to use a WebGL backend for inference in order to save time at the expense of CPU processing power, and also to disable type and correctness checks in favor of performance.

Our average end-to-end inference time is approximately 60-70ms to display the autocomplete dropdown once a user adds a token to the text. This is close to the refresh rate of a screen, so the responsivity is perceived as instantaneous.

8.1.2 Ontology Modifications

We develop several ontologies throughout the course of the development of our system for conditions, symptoms, etc.– these ontologies represent groupings of highly-specific UMLS concepts into categories representing more interpretable medical terminologies. While in the ideal case these ontologies are based on firm clinical knowledge and should remain largely unchanged across use-cases, there are several modes by which a clinician or organization may benefit from a more dynamic ontology structure. For example, an individual doctor may find it more useful to categorize concepts in a way that aligns with their personal medical decision-making process, or at a more global level clinicians may agree that a certain categorization is incorrect or needs to be updated to reflect new clinical knowledge.

To handle such changes, we allow users of our tool to directly request ontology modifications from the client UI. These front-end requests naturally record the proposed change, but also collect the context in which the change was requested. This contextual information includes the section of the note the doctor was working on when the request was made, the clinician making the request, and the patient whom the note was about.

From the back-end, we first immediately modify ontologies at a local level, allowing clinicians to customize the system for themselves to their liking. At a global level, we collect all requested modifications, and surface them at weekly intervals to experts who can curate the changes that would be beneficial if implemented globally. By using the context in which modifications were requested, we can identify parts of clinical notes that may require different levels of granularity or different types of categorization. We can further determine what kinds of patients our system does not categorize well, and improve the ontology accordingly. Thus, while the ontology is mostly invariant, user preferences can quickly be incorporated through local changes. At a global level, the ontology can evolve dynamically to mitigate potential flaws and improve the user experience using human-in-the-loop curated modification.

8.1.3 Collaborative Documentation

EHR notes are often written by multiple clinicians. Doctors sometimes use scribes on certain shifts to note-take while they talk to patients. BIDMC is a teaching hospital, so medical students and ED residents often check in on patients before an attending does. All parties can contribute to the final note.

We create a collaborative editor at the note-level. The clinical server maintains a single lock per user, which the client UI can then acquire and release at will in order to write a patient's note. Users do not need to possess a lock to read notes. This is useful for patient handoff, because doctors can look through pinned notes, tagged chips, and explore any structured information inserted in the note in order

to understand the patient's history and presentation. Links to the patient's medical record can contextualize medical decision making and treatment course.

While we don't need to support multiple authors simultaneously editing the same words or sentence of the note, as this rarely occurs in the ED, our current locking scheme is coarse. Future might work might instead allow a user to edit a single section of the note at a time, and lock at this level instead.

To maintain backwards compatibility, any note in our system can be exported to a plaintext representation that removes any sidebar content. This version of the note can be automatically formulated as a discharge summary and filed in a patient's OMR using BIDMC's current note export system. The workflow is thus minimally interrupted from what is presently used.

8.2 System Feedback & User Metrics

A clinician can use our tool to improve documentation and clinical decision making. However, another goal was to use the data collected with our tool in downstream machine learning applications. Ultimately, we want to create a feedback cycle between our learned models and our interface– documented notes should inform our machine learning models, which in turn improve documentation. To this end, we outline metrics collected with our tool for analysis.

Data Entry: Data entry metrics primarily consist of inserted chips. We use `Draft.js` as our underlying editor framework, which saves each newline-delimited span of text as a block. Thus, metrics for each chip are first aggregated at a block (line) level and then further grouped by section. Each chip is specified by the following:

- Its *section* and *block*, which give positional information.
- An *entity key* which is a private identifier used to differentiate between two otherwise identical chips.

- The insertion *type*– whether the chip added from the autocomplete dropdown, automatically post-recognized after typing, selected from a post-recognition dropdown due to ambiguity in the term, and automatically added as default text in a section.

- The *added text* of the chip, which is the plaintext contained in the chip. For a chip about hypertension, this might be `htn`.

- The *initial text* of the chip, which is the plaintext that we typed in order to tag the chip. If the chip was inserted using the autocomplete dropdown, this is the string that was typed before the term was added. If the chip was inserted using a post recognition or default text mechanism, this is the text of the chip that was recognized to trigger a post-correction.

- The concept type of the inserted chip, which can be a `CONDITION`, `SYMPTOM`, `MEDICATION`, `LAB`, `VITAL`, `PROCEDURE`, or `MACRO`. In the future, we hope to add concept types for any part of a medical record that can be inserted into the note, including OMR notes. Snippets of prior notes can thus be cited and referenced in the editor as a chip.

- The unique identifier of the clinical concept, including its *synonym*, parent concept identifier, and any *aggregates*. Conditions are are normalized to UMLS, so, this might look like `HYPERTENSION - C0020538 - htn`. For a lab/vital value that is aggregated over time, this would be normalized to LOINC include information about the aggregation window and the summary statistic, e.g. `HEMATOCRIT - 20570-8 - HCT - LAST 6 MONTHS - MEAN`.

- The username of the clinician who added the chip to the document, as well as the timestamp it was added. This is used to paint a picture of the note over time, and allows to see which users add what types of chips, and when they are added.

- User interactions with the chips, including timestamps of any clicks. We do not

currently collect hover data for each chip because we view that as too noisy to be a good signal of chip relevance.

- Whether the chip required any *ontology modification*, e.g. the addition/deletion of a clinical concept or its synonyms.

Chip metrics can be used to quantify documentation burden, such as how much redundancy is removed and how many keystrokes are saved by using our tool. It can also be used for clinical named entity recognition and understanding how to aggregate quantitative data to convey information. By taking documented conditions and symptoms, we can also build differential diagnosis tools that predict patient outcomes from clinical concepts.

Sidebar: We also collect metrics from the sidebar. Sidebar metrics include timestamped addition of cards, removal of cards, and click-through data for any relationship shown on the card that is further explored (lab trend visualizations, OMR snippets, etc.). We also capture proxies of card relevance, including what cards are pinned and by whom, and whether OMR note snippets are explored in a new tab and how long the window is in focus. All metrics are augmented with the username of the person who is responsible for generating the metric, as well a timestamp of when the metric was added. These metrics can also be used for downstream ML tasks including supervised approaches to patient record summarization, learning representations of a problem-oriented medical record as in [82], and even better understanding patient handoff by analyzing the utility of sidebar cards in contextualizing medical decision making.

Usability & Learnability: Finally, an important component of our tool is adoption. We hope to use these metrics to quantitatively understand the usability and learnability of the tool. By analyzing documentation patterns from both novice and expert users, we can ascertain how often "superuser" features like macros, slash commands, and lab aggregations are used. We can also visualize how often different

features are accessed by a given user over time to understand the learning curve of the tool. By being systematic in what features we explicitly introduce to users when they are first acquainted with the tool, we can also gauge the discoverability of the tool and how new features are surfaced and interacted with. We hope to report high-level statistics about tool use including the average time it takes to write a note with our system against a simpler one.

We acknowledge that the learnability of the tool is partially governed by how users are onboarded. Inspired by [17], which discusses how to best onboard medical practitioners in collaborative human-ML tools, we seek to make the user conscious of the benefits, limitations, and optimized objectives of our learned algorithms. As such, for any feature with a machine learning component, we discuss what data it has access to, how conservative its predictions are, and any adoption considerations we had in building the tool. Our onboarding procedure is also task-oriented– users are presented with a series of tasks that are posed as questions (e.g. `how can I visualize a patient's lab trend?`) and are provided with written explanations and video screencaptures of the feature in question. Onboarding material can be found here.

Chapter 9

Discussion and Conclusion

EHRs have introduced significant burden on physicians, and to adapt, doctors have resorted to using overloaded jargon that then renders clinical notes unusable for downstream clinical care. The lack of clean labels for unstructured text also inhibits how we can utilize machine learning techniques to transform healthcare. There is a real need to modernize and exploit the information hidden within notes without interrupting the clinical workflow. In this thesis, we outline a novel EHR system with an intelligent user interface. This EHR system fundamentally transforms clinical documentation through the unobtrusive live-tagging of clinical concepts. Tagged concepts enable the synthesis and visualization of prior notes, lab trends, past procedures, related medications, and more. We draw and innovate on on existing literature from clinical summarization, named entity recognition, information retrieval, HCI and beyond to do so.

9.1 Limitations

Our EHR implementation significantly reduces documentation burden on clinicians and allows for improved information retrieval. However, our iteration on this tool was driven by the feedback of BIDMC emergency department physicians, and this

presents the two main practical limitations of our work in terms of deployment.

First, EHRs must be adapted to other clinical specialities. There are several non-trivial distinctions between these specialties – for example, the emergency department is unique in that clinicians face a wide breadth of conditions, but rarely have to consider the deeply longitudinal challenges presented by chronic diseases. While our EHR is generalize and adaptable enough to be translated to other environments in which different kinds of data may be more relevant, this has yet to be realized and as such our system's proven usefulness is limited to the ED. The biggest difference between specialties is the language and vocabulary that is used– we would have to modify our autocomplete ontologies. Missingness and poor concept disambiguation in ontologies can be debilitating for doctors, and while we try to get as much coverage possible, we do support on-the-fly modification of ontologies in our tool. Doctors can suggest ontology edits (adding/editing/deleting a concept or a synonym) while writing a note, which is reflected in their own UI. We plan to regularly update ontologies during initial stages of system use.

Ideally, a robust EHR system would provide a unified but adaptable framework across all departments of a medical institution, and allow for the sharing of information in an intuitive way. In order to provide proper continuity of care, we would also like our system to be able to adapt to the needs of different institutions while still allowing information to flow as well. Our work is potentially limited in this regard as well, as it has only been tested within a single institution. Given the success of this EHR in our limited setting, we hope to ameliorate these limitations through future extensions of our work. The quick iteration on our tool was enabled by BIDMC's custom EHR portal, which is unique to the hospital.

In addition, design choices of our system were driven by limitations with both our dataset and our scope. As an example, the lack any proxy for snippet relevance (such as click-through data) forced us to use a simpler model over something learned end-to-end. This also raises questions of how to formulate mental models of OMR relevance– for note retrieval to be useful, doctors need to save time. Parsing the output

of a learned model is inherently time-consuming because the behavior may not be deterministic or even consistent. Even if we surface snippets that are exactly relevant to the patient's presentation, this does not imply that we are directly benefitting doctors with our current setup.

Finally, while our tool facilitates data collection and annotation at the point-of-care, we hope to be more rigorous about the ways in which we incentivize *high quality data collection*. Adding modifier capture to our data entry is one step towards doing this. By taking a data-driven approach to analyzing gaps in current clinical data, we can make sure the methods and interactions we use to capture data fix these shortcomings as much as possible.

9.2 Future Work

There are several directions in which our EHR system could be improved, and these changes broadly fall into two categories– algorithmic advancements and further development of the user experience for clinicians.

From the algorithmic perspective, we hope to improve relevant metrics throughout our entire data entry and data retrieval pipelines:

- We featurize textual data in our EHR system using a trie-based extraction algorithm for named entity recognition – the data collected by this algorithm is the key driver of all of our downstream models, and improvements in its quality may have significant effects for the overall system. As we note in Table 4.1, more complex NER and text-featurization strategies that have seen success in the broader literature are limited in this application due to issues of latency. However, leveraging more efficient modifications of such algorithms could allow for more thorough data collection. This idea extends to the learning of modifiers, which in our current system is driven by the relatively simple NegEx algorithm for negation detection. Using a more complex model, we could potentially

integrate measures of certainty, negation, polarity and other modifiers of clinical interest as described in Chapter 5.5.1.

- We model the process of data entry, or autocompletion, as a human-augmented natural language model – our choices of model are limited both by data and by computational constraints. As such, we chose to factor our model in such a way that minimizes the need to dynamically use data that is actively being entered by the user. Allowing our autocomplete rankings to be fully dynamic presents a large computational challenges, but the successes of generative language models for autocompletion in other domains [20] indicates that this may nevertheless be a fruitful area for further exploration.

- We hope to further expand the functionality of the sidebar in future work. In particular, risk stratification and differential diagnosis tools could prove invaluable to physicians by providing nuanced learned insights.

We also seek to make the user experience more seamless and integrated. As an example, there may be utility in providing different visualizations for different labs or vital sign measurements, in contrast to the box-and-whiskers plots used for all such numerical values at present, in order to better present information. We further see room for improvement in terms of the high-level goal of building a truly live document, and in future iterations of this tool we hope to include functionality to better incorporate common template-like paradigms of medical actions. For instance, doctors might be able to define macros which would automatically expand and template out fields for information we now know will be collected in the future – if the macro `@hct` orders a hematocrit lab for the patient, for example, we would create a empty box in the text that is filled in automatically when the lab result is ready. We finally seek to give clinicians and other parties of interest a better high-level view of the overall health of all patients, or a specific group of patients. While we have developed tools to convey a precise sense of a particular patient's health, a more comparative or summarative perspective is useful as well.

Lastly, our EHR builds a system from the ground up, relying on careful retroactive featurization of data collected using legacy note-taking utilities. As clinicians use our system, it will automatically curate and aggregate large amounts of labeled EHR data. This can then be used to further improve existing machine-learning algorithms used to collect and surface data, and to inspire new applications as sufficient high-quality data becomes available. It is key that future implementations of this tool incorporate a feedback loop in which the data collected by the tool drives algorithmic improvements, which can also be used to identify longitudinal data drift.

The methods outlined in this thesis, in tandem with the ideas presented for future work, can truly revolutionize clinical documentation for doctors, patients, and algorithms. By integrating machine learning methodologies into documentation practices, we can usher in a new era of EHRs that assist rather than impede physicians.

Appendix A

Appendix

A.1 NegEx Algorithm

We use a version of the NegEx algorithm [19] in order to perform a rule-based negation detection on clinical text. The algorithm greedily iterates through words in a piece of text and assigns them to a negated context if they are preceded by predefined keyword triggers.

Pseudocode for the algorithm is shown below in Figure A-1.

A.2 Examples of Clinical Notes

Here, we show examples of a triage note, chief complaint, patient vitals, and a clinician note. To preserve patient privacy, these examples are fake, but mimic the formatting and style of real data.

Triage Note

```
pt with ruq abd pain and nonproductive cough
```

Chief Complaint

```
fullstops = ['.', '-', ';']
midstops = ['+', 'but', 'and', 'pt', '.', ';', 'except', 'reports', '
    alert', 'complains', 'has', 'states', 'secondary', 'per', 'did', '
    other', 'p/w', 'presents', 'presenting', 'presented', ':']
negwords = ['no', 'not', 'denies', 'without', 'non', 'lack']

def negation_detection(words):
    flag = 0
    res = []
    for i, w in enumerate(words):
        neg_start_condition = (flag == 1)
        neg_stop_condition =  (w in fullstops + midstops + negwords) or
    (i > 0 and words[i-1][-1] in (fullstops + ['\n']))
        neg_end_of_list = (i==(len(words)-1) )
        if neg_start_condition and neg_stop_condition:
            flag = 0
            res += [(start_index, i-1)]
        elif neg_start_condition and neg_end_of_list:
            flag = 0
            res += [(start_index, i)]
        if w in negwords:
            flag = 1
            start_index = i
    return res
```

Figure A-1: Pseudocode of the rule-based negation detection algorithm.

```
ruq abd pain
```

Vitals

```
Blood Pressure: 140/90 mmHg
Heart Rate: 109 BPM
Pain: 8 (out of 10)
Sex: F
Age: 66
Respiratory Rate: 92%
Temperature: 99 (deg. Fahrenheit)
Pulse Oxygen (Oxygen Saturation): 96
```

Clinical Note

```
HPI: 66 y/o F p/w ruq abd pain and nonproductive cough.
```

No fever, nausea, or chills.
History of chronic abdominal pain over last 4-5 years,
as well as htn and dmii.

PMH: htn, dmii, chronic abdominal pain, hysterectomy in 2004

MEDICATIONS: metoprolol tartrate, metformin

FAMILY HISTORY: Diabetes in mother,
father (deceased) hypertensive

SOCIAL HISTORY: no smoking, drinks socially

REVIEW OF SYSTEMS:
Constitutional - no fever, chills, nausea
Head / Eyes - no diplopia
ENT - no earache
Resp - nonproductive cough, mild
Cards - no chest pain
Abd - ruq abd pain
Flank - no dysuria
Skin - no rash
Ext - no back pain
Neuro - no headache
Psych - no depression

PHYSICAL EXAM: Ruq abd pain, tender to touch,
with some bloating.

MDM:

```
66 y/o F p/w ruq abd pain and mild cough. She reports
she had a cold last week, so cough
is likely symptom of that.

Epigastric pain with mild bloating and minor
heartburn. Gave an antacid to relieve pain.

Glucose levels are elevated compared to baseline
(140 6 hours ago, 120 averaged over last six months).
Says she will work on controlling diet more.

DIAGNOSIS: epigastric pain/heartburn
```

A.3 OMR Annotation Tool

The OMR annotation tool alluded to in Chapter 6 is a prototype of a data collection tool for clinicians to highlight and annotate relevant snippets of the OMR. This could be used to eventually curate a large-scale dataset of pieces of the medical record that are important in treating and diagnosing a patient, which would be an invaluable resource for the clinical informatics community.

The tool is shown below in Figure A-2. It is a Flask/JavaScript application. On the lefthand panel, the chief complaint, triage assessment, list of medications, and early sections of the clinical note are displayed. The middle pane contains a view of various OMR notes in chronological order, which can be panned through with the blue arrows, or by using the exact stringmatch keyword search in the righthand panel. Sections of the OMR note that are relevant can be annotated simply by highlighting the section with the cursor and hitting enter. Then, the character spans of each annotation are saved with the annotations are submitted.

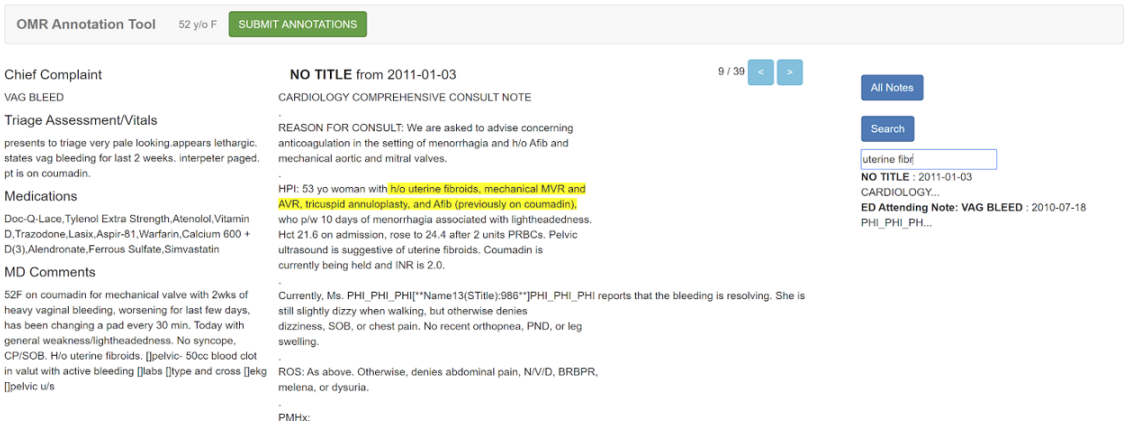


Figure A-2: OMR Annotation Tool, which allows the user to read through, search, and highlight OMR data in the context of a particular ED visit.

Hyperparameter	Values Tested
Input sequence length	128, 256, 512
Kernel sizes	`[1, 1, 1], [2, 2, 2], [1, 2, 3, 5, 7], [2, 4, 6], [1, 3, 5, 7], [5, 5, 5], [1, 2, 3, 4]`
Number of filters per kernel size	25, 50, 100, 200
Embedding sizes	64, 128, 256

Table A.1: Hyperparameters for tuning deep models for differential diagnosis. Model class is a multi-branch one-dimension CNN trained to make a multiclass prediction; see Section 7.4.1 for architectural details.

[This patient is a 68 year old male who complains of a swallowed half denture. The denture was broken, and while the patient was eating lunch, it broke off and he swallowed it. The patient reports no discomfort or pain. He is handling secretions well.]

[Timing: Sudden
Onset:
Quality: swallowed denture
Duration: few hours
Location: GI tract, stomach, esophagus
Signs/symptoms: handline secretions well. No pain.]

[PAST MEDICAL HISTORY
PMH includes a CABG 5 years ago. DM as well.
[Medications: Coumadin, Crestor, metoprolol succinate, metformin, lisinopril. Allergies and Reactions: NKDA.]
Social History: +alcohol, +smoking]

[REVIEW OF SYSTEMS-- All other systems reviewed and negative.]

[PHYSICAL EXAMINATION
Constitutional: Comfortable
HEENT: Normocephalic, atraumatic
Chest: Clear to auscultation
[Cardiovascular: Regular Rate and Rhythm, Normal first and
second heart sounds]
Abdominal: Soft, Nontender, Nondistended, obese
Neuro: Speech fluent
Psych: Normal mood, Normal mentation]

[RADIOLOGY

Note(s): Soft tissue films of the neck-no foreign body identified
chest x-ray-no foreign body identified
abdominal film-the partial denture is visualized in the small bowel]

[MEDICAL DECISION MAKING

Prior to my seeing him the patient had been ordered for soft tissue films of the neck. There is no foreign body visualized on these. Patient had a chest x-ray and a KUB to identify the location of his denture. [The denture has already passed through the stomach and is in the small bowel.] We spoke to surgery who reviewed the patient's films and felt that he should not have any problems passing the denture. The patient has no pain. He has been discharged
home with expectant management.
Service Consulted at 00:20 Surgery Final ED Diagnosis 1: Small bowel foreign body]

[This uploaded version of the chart may not be the final one;
some addenda and test results may not be entered into this
OMR note.]

Figure A-3: Candidate OMR snippets created with heuristics described in Section 6.2. Colored [ ] characters denote each (possibly overlapping) snippet span.

Bibliography

[1] Shashank Agarwal and Hong Yu. Detecting hedge cues and their scope in biomedical text with conditional random fields. *Journal of Biomedical Informatics*, 43(6):953–961, December 2010.

[2] Muhammad Ahmad, Carly Eckert, Greg McKelvey, Kiyana Zolfagar, Anam Zahid, and Ankur Teredesai. Death vs. Data Science: Predicting end of life. 2018.

[3] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An Aid to Bibliographic Search. *Commun. ACM*, 18(6):333–340, June 1975.

[4] Hanan Aldarmaki, Mahesh Mohan, and Mona Diab. Unsupervised word mapping using structural similarities in monolingual embeddings. *Transactions of the Association for Computational Linguistics*, 6:185–196, 2018.

[5] Ilseyar Alimova and Elena Tutubalina. Multiple features for clinical relation extraction: a machine learning approach. *Journal of Biomedical Informatics*, 103:103382, 02 2020.

[6] Duaa Aljabri, Adrian Dumitrascu, Caroline Burton, Launia White, Mahmud Khan, Sudha Xirasagar, Ronnie Horner, and James Naessens. Patient portal adoption and use by hospitalized cancer patients: a retrospective study of its impact on adverse events, utilization, and patient satisfaction. *BMC Medical Informatics and Decision Making*, 18(1):70, 2018.

[7] Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. Text summarization techniques: A brief survey. *CoRR*, abs/1707.02268, 2017.

[8] Diane Alonso, Anne Rose, Catherine Plaisant, and Kent Norman. Viewing personal history records: A comparison of tabular format and graphical presentation using LifeLines. *Behaviour and Information Technology*, 17, 12 1998.

[9] Emily Alsentzer and Anne Kim. Extractive Summarization of EHR discharge notes. *CoRR*, abs/1810.12085, 2018.

[10] Chrissanthi Angeli. Diagnostic expert systems: From expert's knowledge to real-time systems. 2010.

[11] Joan Ash, Marc Berg, and Enrico Coiera. Some unintended consequences of information technology in health care: The nature of patient care information system-related errors. *Journal of the American Medical Informatics Association*, 11:104–12, 03 2004.

[12] G. Octo Barnett, James J. Cimino, John A. Hupp, and E. P. Hoffer. DXplain. An evolving diagnostic decision-support system. *Journal of the American Medical Informatics Association*, 258(1):67–74, Jul 1987.

[13] Andrew L. Beam, Benjamin Kompa, Inbar Fried, Nathan P. Palmer, Xu Shi, Tianxi Cai, and Isaac S. Kohane. Clinical Concept Embeddings Learned from Massive Sources of Medical Data. *CoRR*, abs/1804.01486, 2018.

[14] Robert Bill, Serguei Pakhomov, Elizabeth S. Chen, Tamara J. Winden, Elizabeth W. Carter, and Genevieve B. Melton. Automated extraction of family history information from clinical notes. *AMIA Annual Symposium Proceedings*, 2014:1709–1717, 2014.

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(null):993–1022, March 2003.

[16] Andrew D. Boyd, Christine D. Young, Margret Amatayakul, Michael G. Dieter, and Lawrence M. Pawola. Developing Visual Thinking in the Electronic Health Record. *Studies in Health Technology and Informatics*, 245:308–312, 2017.

[17] Carrie J. Cai, Samantha Winter, David Steiner, Lauren Wilcox, and Michael Terry. "Hello AI": Uncovering the Onboarding Needs of Medical Practitioners for Human-AI Collaborative Decision-Making. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019.

[18] Pascale Carayon, Tosha Wetterneck, Bashar Alyousef, Roger Brown, Randi Cartmill, Kerry McGuire, Petter Hoonakker, Jason Slagle, Kara Roy, James Walker, Matthew Weinger, Anping Xie, and Kenneth Wood. Impact of electronic health record technology on the work and workflow of physicians in the intensive care unit. *International Journal of Medical Informatics*, 84, 04 2015.

[19] Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*, 34(5):301–310, Oct 2001.

[20] Mia Xu Chen, Benjamin N. Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, Timothy Sohn, and Yonghui Wu. Gmail Smart Compose: Real-Time Assisted Writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '19, page 2287–2295, New York, NY, USA, 2019. Association for Computing Machinery.

[21] Yan Chen, Huanying Gu, Yehoshua Perl, and James Geller. Structural group-based auditing of missing hierarchical relationships in UMLS. *Journal of biomedical informatics*, 42:452–67, 09 2008.

[22] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[23] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. Gram: Graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 787–795, New York, NY, USA, 2017. Association for Computing Machinery.

[24] Tom Christensen and Anders Grimsmo. Instant availability of patient records, but diminished availability of information: a multi-method study of gps use of electronic patient records. *BMC medical informatics and decision making*, 8:12, 02 2008.

[25] Imke Christiaans, Klaartje van Engelen, Irene M. van Langen, Erwin Birnie, Gouke J. Bonsel, Perry M. Elliott, and Arthur A.M. Wilde. Risk stratification for sudden cardiac death in hypertrophic cardiomyopathy: systematic review of clinical risk markers. *EP Europace*, 12(3):313–321, 01 2010.

[26] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? An Analysis of BERT's Attention. *CoRR*, abs/1906.04341, 2019.

[27] Genna Cohen, Charles Friedman, Andrew Ryan, Caroline Richardson, and Julia Adler-Milstein. Variation in Physicians' Electronic Health Record: Documentation and Potential Patient Harm from That Variation. *Journal of General Internal Medicine*, 34:1–13, 06 2019.

[28] Michelle A. Cretikos, Rinaldo Bellomo, Ken Hillman, Jack Chen, Simon Finfer, and Arthas Flabouris. Respiratory rate: the neglected vital sign. *Med. J. Aust.*, 188(11):657–659, Jun 2008.

[29] Jiangbo Dang, Amir Hedayati, Ken Hampel, and Candemir Toklu. An ontological knowledge framework for adaptive medical workflow. *Journal of Biomedical Informatics*, 41(5):829–836, October 2008.

[30] James Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In

Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[31] N.L. Downing, David W. Bates, and Christopher A. Longhurst. Physician Burnout in the Electronic Health Record Era: Are We Ignoring the Real Cause? *Annals of Internal Medicine*, 169(1):50–51, 07 2018.

[32] Noémie Elhadad, Min-Yen Kan, Judith L. Klavans, and Kathleen McKeown. Customization in a unified framework for summarizing medical literature. *Artificial Intelligence in Medicine*, 33 2:179–98, 2005.

[33] Noémie Elhadad, Kathleen McKeown, David R. Kaufman, and Desmond A. Jordan. Facilitating physicians' access to information via tailored text summarization. *AMIA Annual Symposium Proceedings*, pages 226–30, 2005.

[34] Noémie Elhadad, Sharon Lipsky Gorman, Jamie S. Hirsch, Connie Liu, David K. Vawdrey, and Marc Sturm. Harvest, a holistic patient record summarizer at the point of care. In *AMIA*, 2014.

[35] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 213–220, New York, NY, USA, 2008. Association for Computing Machinery.

[36] J. Michael Fitzmaurice, Karen Adams, and John M. Eisenberg. Medical informatics support at the agency for healthcare research and quality. *Journal of the American Medical Informatics Association : JAMIA*, 9 2:144–60, 2002.

[37] Mindy E Flanagan, Emily S Patterson, Richard M Frankel, and Bradley N Doebbeling. Evaluation of a physician informatics tool to improve patient handoffs. *Journal of the American Medical Informatics Association*, 16(4):509–515, 2009.

[38] Nigel Jie Ming Fong. *Algorithms in Differential Diagnosis: How to Approach Common Presenting Complaints in Adult Patients, for Medical Students and Junior Doctors*. World Scientific, 2018.

[39] Rebekah L. Gardner, Emily Cooper, Jacqueline Haskell, Daniel A. Harris, Sara Poplau, Philip J. Kroth, and Mark Linzer. Physician stress and burnout: the impact of health information technology. *Journal of American Medical Informatics Association*, 26(2):106–114, Feb 2019.

[40] Sebastian Gehrmann, Franck Dernoncourt, Yeran Li, Eric T. Carlson, Joy T. Wu, Jonathan Welt, John Foote, Edward T. Moseley, David W. Grant, Patrick D. Tyler, and Leo Anthony Celi. Comparing deep learning and concept extraction based methods for patient phenotyping from clinical narratives. *PLoS ONE*, 13, 2018.

[41] Macda Gerard, Hannah Chimowitz, Alan Fossa, Fabienne Bourgeois, Leonor Fernandez, and Sigall K. Bell. The importance of visit notes on patient portals for engaging less educated or nonwhite patients: Survey study. *J Med Internet Res*, 20(5):e191, May 2018.

[42] Debora Goetz Goldberg, Anton J. Kuzel, Lisa Bo Feng, Jonathan P. DeShazo, and Linda E Love. EHRs in primary care practices: benefits, challenges, and successful strategies. *The American Journal of Managed Care*, 18 2:e48–54, 2012.

[43] Yoav Goldberg. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287, 2019.

[44] Bruce Goldman. Green Button: The promise of personalizing medical practice guidelines in real time, 2015.

[45] Jen J. Gong and John V. Guttag. Learning to summarize Electronic Health Records using Cross-Modality Correspondences. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, volume 85 of *Proceedings of Machine Learning Research*, pages 551–570, Palo Alto, California, 17–18 Aug 2018. PMLR.

[46] Sergey Goryachev, Hyeoneui Kim, and Qing Zeng-Treitler. Identification and extraction of family history information from clinical reports. *AMIA Annual Symposium Proceedings*, pages 247–251, Nov 2008.

[47] Nathaniel R. Greenbaum, Yacine Jernite, Yoni Halpern, Shelley Calder, Larry A. Nathanson, David Sontag, and Steven Horng. Contextual autocomplete: A novel user interface using machine learning to improve ontology usage and structured data capture for presenting problems in the emergency department. *bioRxiv*, 2017.

[48] Yoni Halpern, Steven Horng, Youngduck Choi, and David Sontag. Electronic medical record phenotyping using the anchor and learn framework. *Journal of the American Medical Informatics Association*, 23(4):731–740, 04 2016.

[49] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[51] William Hsu, Ricky Taira, Suzie El-Saden, Hooshang Kangarloo, and Alex Bui. Context-based electronic health record: Toward patient specific healthcare. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 16:228–34, 03 2012.

[52] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission. *CoRR*, abs/1904.05342, 2019.

[53] Sarthak Jain and Byron C. Wallace. Attention is not explanation. *CoRR*, abs/1902.10186, 2019.

[54] Stephen B Johnson, Suzanne Bakken, Daniel Dine, Sookyung Hyun, Eneida Mendonça, Frances Morrison, Tiffani Bright, Tielman Van Vleck, Jesse Wrenn, and Peter Stetson. An Electronic Health Record based on structured narrative. *Journal of the American Medical Informatics Association*, 15(1):54–64, 2008.

[55] Johanna Kaipio, Hannele Hyppönen, Tinja Lääveri, Jukka Vänskä, Jarmo Reponen, and Ilkka Winblad. National questionnaire study on clinical ict systems proofs: Physicians suffer from poor usability. *International journal of medical informatics*, 80:708–25, 07 2011.

[56] Johanna Kaipio, Tinja Lääveri, Hannele Hyppönen, Suvi Vainiomäki, Jarmo Reponen, André Kushniruk, Elizabeth M. Borycki, and Jukka Vänskä. Usability problems do not heal by themselves: National survey on physicians' experiences with EHRs in Finland. *International Journal of Medical Informatics*, 97:266–281, 2017.

[57] Richard Kamin, Thomas Nowicki, David Courtney, and Robert Powers. Pearls and Pitfalls in the Emergency Department Evaluation of Abdominal Pain. *Emergency medicine clinics of North America*, 21:61–72, vi, 03 2003.

[58] Alok A. Khorana. Physician as typist. *Journal of Clinical Oncology*, 28(24):3899–3900, 2010. PMID: 20547988.

[59] Susan Koch-Weser, William Dejong, and Rima E. Rudd. Medical word use in clinical encounters. *Health expectations : an international journal of public participation in health care and health policy*, 12(4):371–382, Dec 2009. HEX555[PII].

[60] Ross Koppel, Joshua P. Metlay, Abigail Cohen, Brian Abaluck, A. Russell Localio, Stephen E. Kimmel, and Brian L. Strom. Role of Computerized Physician Order Entry Systems in Facilitating Medication Errors. *Journal of the American Medical Informatics Association*, 293(10):1197–1203, 03 2005.

[61] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT, 2019.

[62] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[63] Andre Kushniruk, Marc M. Triola, Elizabeth M. Borycki, Ben P. Stein, and Joseph L. Kannry. Technology induced error and usability: The relationship between usability problems and prescription errors when using a handheld application. *International journal of Medical Informatics*, 74 7-8:519–26, 2005.

[64] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics.

[65] Archana Laxmisan, Allison McCoy, Adam Wright, and Dean Sittig. Clinical summarization capabilities of commercially-available and internally-developed electronic health records. *Applied clinical informatics*, 3:80–93, 02 2012.

[66] Neal Lewis, Daniel Gruhl, and Hui Yang. Extracting family history diagnosis from clinical texts. In *BICoB*, 2011.

[67] Fei Li and Hong Yu. ICD Coding from Clinical Text using Multi-Filter Residual Convolutional Neural Network. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[68] Jennifer Liang, Ching-Huei Tsou, and Ananya Poddar. A novel system for extractive clinical note summarization using EHR data. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 46–54, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[69] Katherine Liao, Tianxi Cai, Guergana Savova, Shawn Murphy, Elizabeth Karlson, Ashwin Ananthakrishnan, Vivian Gainer, Stanley Shaw, Zongqi Xia, Peter Szolovits, Susanne Churchill, and Isaac Kohane. Development of phenotype algorithms using electronic medical records and incorporating natural language processing. *BMJ*, 350:h1885–h1885, 04 2015.

[70] Steven Y. Lin, Tait D. Shanafelt, and Steven M. Asch. Reimagining Clinical Documentation with Artificial Intelligence. *Mayo Clinic Proceedings*, 93(5):563–565, May 2018.

[71] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany, August 2016. Association for Computational Linguistics.

[72] Peter J. Liu. Learning to write notes in electronic health records. *CoRR*, abs/1808.02622, 2018.

[73] Yuan Liu, Ayush Jain, Clara Eng, David H. Way, Kang Young Lee, Peggy Bui, Kimberly Kanada, Guilherme de Oliveira Marinho, Jessica Castro Gallegos, Sara Gabriele, Vishakha Gupta, Nalini Singh, Vivek S. Natarajan, Rainer Hofmann-Wellenhof, Greg S Corrado, Lily H. Peng, Dale W. Webster, Dennis Ai, Susan J. Huang, Yun Liu, R C Dunn, and David Coz. A deep learning system for differential diagnosis of skin diseases. *Nature Medicine*, 26:900 – 908, 2020.

[74] Yen-Fu Luo, Weiyi Sun, and Anna Rumshisky. MCN: A comprehensive corpus for medical concept normalization. *Journal of Biomedical Informatics*, 92:103132, 02 2019.

[75] Fenglong Ma, Quanzeng You, Houping Xiao, Radha Chitta, Jing Zhou, and Jing Gao. Kame: Knowledge-based attention model for diagnosis prediction in healthcare. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 743–752, New York, NY, USA, 2018. Association for Computing Machinery.

[76] Susanna Martikainen, Johanna Viitanen, Mikko Korpela, and Tinja Lääveri. Physicians' experiences of participation in healthcare IT development in Finland: Willing but not able. *International Journal of Medical Informatics*, 81 2:98–113, 2012.

[77] Susana Martins, Yuval Shahar, Maya Galperin, Herbert Kaizer, Dina Goren Bar, Deborah McNaughton, Lawrence Basso, and Mary Goldstein. Evaluation of KNAVE-II: A tool for intelligent query and exploration of patient data. *Studies in Health Technology and Informatics*, 107:648–52, 02 2004.

[78] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[79] Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C. Schmidt, Hongfang Liu, and Mathew Palakal. Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of Biomedical Informatics*, 54, 03 2015.

[80] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[81] Randolph A Miller, Harry E Pople Jr, and Jack D Myers. Internist-I, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307(8):468–476, 1982.

[82] James Mullenbach, Jordan Swartz, T. Greg McKelvey, Hui Dai, and David Sontag. Knowledge base completion for constructing problem-oriented medical records. *ArXiv*, abs/2004.12905, 2020.

[83] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable Prediction of Medical Codes from Clinical Text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[84] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. ScispaCy: Fast and robust models for biomedical natural language processing. *CoRR*, abs/1902.07669, 2019.

[85] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann an imprint of Academic Press, a Harcourt Science and Technology Company, 1993.

[86] Stephen G. Pauker, G. Anthony Gorry, Jerome P. Kassirer, and William B. Schwartz. Towards the simulation of clinical cognition: Taking a present illness by computer. *The American Journal of Medicine*, 60 7:981–96, 1976.

[87] Yifan Peng, Xiaosong Wang, Le Lu, Mohammadhadi Bagheri, Ronald M. Summers, and Zhiyong Lu. Negbio: a high-performance tool for negation and uncertainty detection in radiology reports. *CoRR*, abs/1712.05898, 2017.

[88] Roy Perlis. A clinical risk stratification tool for predicting treatment resistance in major depressive disorder. *Biological psychiatry*, 74, 02 2013.

[89] David Pieczkiewicz, Stanley Finkelstein, and Marshall Hertz. Design and evaluation of a web-based interactive visualization system for lung transplant home monitoring data. *AMIA Annual Symposium Proceedings*, 2007:598–602, 02 2007.

[90] R. Pivovarov, Y. J. Coppleson, S. L. Gorman, D. K. Vawdrey, and N. Elhadad. Can Patient Record Summarization Support Quality Metric Abstraction? In *AMIA*, volume 2016, pages 1020–1029, 2016.

[91] Rimma Pivovarov and Noémie Elhadad. Automated methods for the summarization of electronic health records. *Journal of the American Medical Informatics Association : JAMIA*, 22, 04 2015.

[92] Peter J. Pronovost, Sean M. Berenholtz, Todd Dorman, Pam A Lipsett, Terri Simmonds, and Carol Haraden. Improving communication in the ICU using daily goals. *Journal of Critical Care*, 18 2:71–5, 2003.

[93] Yair G. Rajwan and George R. Kim. Medical information visualization conceptual model for patient-physician health communication. In *Proceedings of the*

1st ACM International Health Informatics Symposium, IHI '10, page 512–516, New York, NY, USA, 2010. Association for Computing Machinery.

[94] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, Singapore, August 2009. Association for Computational Linguistics.

[95] Padmanabhan Ramnarayan, Amanda Tomlinson, Gautam Kulkarni, Anupama Rao, and Joseph Britto. A novel diagnostic aid (ISABEL): development and preliminary evaluation of clinical performance. *Studies in Health Technolology and Informatics*, 107(Pt 2):1091–1095, 2004.

[96] Murali Ravuri, Anitha Kannan, Geoffrey J. Tso, and Xavier Amatriain. Learning from the experts: From expert systems to machine learned diagnosis models. *CoRR*, abs/1804.08033, 2018.

[97] Alexander Rind, Taowei D. Wang, Wolfgang Aigner, Silvia Miksch, Krist Wongsuphasawat, Catherine Plaisant, and Ben Shneiderman. *Interactive Information Visualization to Explore and Query Electronic Health Records*. 2013.

[98] J. Rogers, C. Puleston, and A. Rector. The clef chronicle: Patient histories derived from electronic health records. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pages x109–x109, 2006.

[99] Ruth Reátegui Rojas and Sylvie Ratté. Comparison of metamap and ctakes for entity extraction in clinical notes. *BMC Medical Informatics and Decision Making*, 18, 2018.

[100] S. Rosenbloom, Edward Shultz, and Adam Wright. Managing the flood of codes: maintaining patient problem lists in the era of meaningful use and icd10. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, 2012:8–10, 11 2012.

[101] S Trent Rosenbloom, Randolph A Miller, Kevin B Johnson, Peter L Elkin, and Steven H Brown. Interface terminologies: facilitating direct entry of clinical data into electronic health record systems. *Journal of the American Medical Informatics Association*, 13(3):277–288, 2006.

[102] Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. Learning a health knowledge graph from electronic medical records. *Scientific Reports*, 7(1):5994, 2017.

[103] Adam Rule, Isaac H. Goldstein, Michael F. Chiang, and Michelle R. Hribar. Clinical documentation as end-user programming. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[104] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[105] Guergana K. Savova, James J. Masanz, Philip V. Ogren, Jiaping Zheng, Sunghwan Sohn, Karin Kipper Schuler, and Christopher G. Chute. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association : JAMIA*, 17 5:507–13, 2010.

[106] Jeffrey Schnipper, Jeffrey Linder, Matvey Palchuk, Jonathan Einbinder, Qi Li, Anatoly Postilnik, and Blackford Middleton. Smart Forms in an Electronic Medical Record: Documentation-based clinical decision support to improve disease management. *Journal of the American Medical Informatics Association : JAMIA*, 15:513–23, 04 2008.

[107] Fred Schulte and Erika Fry. Death by 1,000 clicks: Where electronic health records went wrong. *Kaiser Health News*, Jun 2019.

[108] Mark Sharp. Toward a comprehensive drug ontology: Extraction of drug-indication relations from diverse information sources. *Journal of Biomedical Semantics*, 8, 12 2017.

[109] Chaitanya Shivade, Marie-Catherine de Marneffe, Eric Fosler-Lussier, and Albert M. Lai. Extending negex with kernel methods for negation detection in clinical text. In *Proceedings of the Second Workshop on Extra-Propositional Aspects of Meaning in Computational Semantics (ExProM 2015)*, pages 41–46, Denver, Colorado, June 2015. Association for Computational Linguistics.

[110] Catherine Smith, Scott Hetzel, Prudence Dalrymple, and Alla Keselman. Beyond readability: Investigating coherence of clinical text for consumers. *Journal of medical Internet research*, 13:e104, 10 2011.

[111] Illés Solt, Domonkos Tikk, Viktor Gál, and Zsolt Kardkovács. Semantic classification of diseases in discharge summaries using a context-aware rule-based classifier. *Journal of the American Medical Informatics Association : JAMIA*, 16:580–4, 05 2009.

[112] Irena Spasic and Goran Nenadic. Clinical text data in machine learning: Systematic review. *JMIR Medical Informatics*, 8(3):e17984, Mar 2020.

[113] Justin Starren and Stephen B. Johnson. An Object-oriented Taxonomy of Medical Data Presentations. *Journal of the American Medical Informatics Association*, 7(1):1–20, 01 2000.

[114] William W. Stead, Randolph A. Miller, Mark A. Musen, and William R. Hersh. Integration and Beyond: Linking Information from Disparate Sources and into Workflow. *Journal of the American Medical Informatics Association*, 7(2):135–145, 03 2000.

[115] Andrew B Symons and Robert H Seller. *Differential Diagnosis of Common Complaints*. Elsevier Health Sciences, 2017.

[116] H. J. Tange, A. Hasman, P. F. de Vries Robbe, and H. C. Schouten. Medical narratives in electronic medical records. *International Journal of Medical Informatics*, 46(1):7–29, Aug 1997.

[117] Michael Tutty, Lindsey Carlasare, Stacy Lloyd, and Christine Sinsky. The complex case of EHRs: examining the factors impacting the EHR user experience. *Journal of the American Medical Informatics Association*, 26:673–677, 07 2019.

[118] Erik M van Mulligen, H Stam, and Astrid M van Ginneken. Clinical data entry. In *Proceedings of the AMIA Symposium*, page 81. American Medical Informatics Association, 1998.

[119] Tielman T. Van Vleck, Adam Wilcox, Peter D. Stetson, Stephen B. Johnson, and Noémie Elhadad. Content and structure of clinical problem lists: a corpus analysis. *AMIA Annual Symposium Proceedings*, pages 753–757, Nov 2008.

[120] David Vawdrey. Assessing usage patterns of electronic clinical documentation templates. *Journal of the American Medical Informatics Association : JAMIA*, 2008:758–62, 02 2008.

[121] Taowei David Wang, Krist Wongsuphasawat, Catherine Plaisant, and Ben Shneiderman. Visual Information Seeking in Multiple Electronic Health Records: Design recommendations and a process model. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, page 46–55, New York, NY, USA, 2010. Association for Computing Machinery.

[122] J.M. Weis and P.C. Levy. Copy, paste, and cloned notes in electronic health records. *Chest*, 145(3):632–638, 2014.

[123] Jesse O Wrenn, Daniel M Stein, Suzanne Bakken, and Peter D Stetson. Quantifying clinical narrative redundancy in an electronic health record. *Journal of the American Medical Informatics Association*, 17(1):49–53, 2010.

[124] Stephen Wu, Kirk Roberts, Surabhi Datta, Jingcheng Du, Zongcheng Ji, Yuqi Si, Sarvesh Soni, Qiong Wang, Qiang Wei, Yang Xiang, Bo Zhao, and Hua Xu. Deep learning in clinical natural language processing: a methodical review. *Journal of American Medical Informatics Association*, 27(3):457–470, Mar 2020.

[125] Zhang Yijia, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong lu. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific Data*, 6, 12 2019.

[126] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*,

pages 1753–1762, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[127] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.