Fast, Structured Clinical Documentation via Contextual Autocomplete

Divya Gopinath DIVYAGOP@MIT.EDU
Department of Electrical Engineering & Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA

Monica Agrawal MAGRAWAL@MIT.EDU
Department of Electrical Engineering & Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA

Luke Murray LSMURRAY@MIT.EDU
Department of Electrical Engineering & Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA

Steven Horng SHORNG@BIDMC.HARVARD.EDU
Center for Healthcare Delivery Science
Beth Israel Deaconess Medical Center
Boston, MA, USA

David Karger KARGER@MIT.EDU
Department of Electrical Engineering & Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA

David Sontag DSONTAG@MIT.EDU
Department of Electrical Engineering & Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA

Abstract

We present a system that uses a learned autocompletion mechanism to facilitate rapid creation of semi-structured clinical documentation. We dynamically suggest relevant clinical concepts as a doctor drafts a note by leveraging features from both unstructured and structured medical data. By constraining our architecture to shallow neural networks, we are able to make these suggestions in real time. Furthermore, as our algorithm is used to write a note, we can automatically annotate the documentation with clean labels of clinical concepts drawn from medical vocabularies, making notes more structured and readable for physicians, patients, and future algorithms. To our knowledge, this system is the only machine learning-based documentation utility for clinical notes deployed in a live hospital setting, and it reduces keystroke burden of clinical concepts by 67% in real environments.

1. Introduction

Clinicians currently spend more time documenting information in electronic health records (EHRs) than communicating with patients, and the timesink in using inefficient EHRs is posited to be a leading cause of physician stress and burnout (Carayon et al., 2015; Gardner et al., 2019). Doctors prefer using natural language and free-text for documentation over restrictive structured forms (Khorana, 2010), but clinicians have adapted to time-intensive note-writing by relying on overloaded acronyms and jargon (Smith et al., 2011). As an example, consider this sentence from a real Emergency Department (ED) clinical note: `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 multiple sclerosis`.

Consequently, medical documentation is often noisy, ambiguous, and incomplete. The lack of structure in notes further hinders understandability for patients, other physicians, and machines (Aljabri et al., 2018; Gerard et al., 2018; Koch-Weser et al., 2009). The information within EHR notes remains largely untapped and, at present, cannot be easily used for downstream medical care or for machine learning models that rely on structured data.

Statement of Contributions We propose a method called *contextual autocomplete*, which quickly captures clinical concepts at the point-of-care via learned suggestions. To do so, we build a hierarchical language model for clinical concepts that can operate in the noisy domain of ED notes. Our model is designed to be deployed in a live hospital environment, with inputs constrained to the triage information and past medical notes available to a doctor *before* a note is written. While these constraints make it infeasible to build a generative language model, we generalize to the task of *autocompletion*, where we make multiple suggestions for the next clinical concept to document and allow the clinician to determine the correct choice. As all suggestions are mapped to standardized clinical vocabularies, we can simultaneously impose structure on notes as they are being written, disambiguate between concepts, and make documentation faster for clinicians in real-world hospital settings.

Clinical Relevance We present contextual autocompletion as the cornerstone of an intelligent EHR in Figure 1. The contextual autocompletion tool reduces the amount of text a clinician has to type by suggesting relevant terms using a learned context. Tagged 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 clinician types– for example, a tagged condition in an earlier part of a note can be automatically appended to the Past Medical History section that appears later on, as in the right panel of Figure 1. This mitigates the “death by a thousand clicks” phenomenon that EHRs suffer from (Schulte and Fry, 2019). In addition, live-tagging clinical concepts can 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 en-

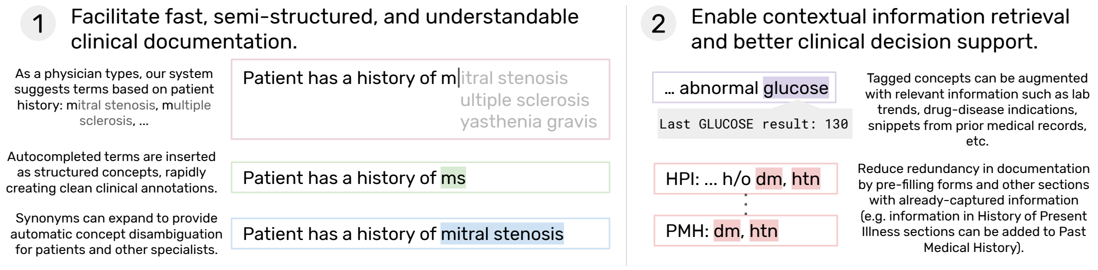

1
Facilitate fast, semi-structured, and understandable clinical documentation.
As a physician types, our system suggests terms based on patient history: mitral stenosis, multiple sclerosis, ...
Patient has a history of m|itral stenosis
ultiple sclerosis
yasthenia gravis
Autocompleted terms are inserted as structured concepts, rapidly creating clean clinical annotations.
Patient has a history of ms
Synonyms can expand to provide automatic concept disambiguation for patients and other specialists.
Patient has a history of mitral stenosis
2
Enable contextual information retrieval and better clinical decision support.
... abnormal glucose
Last GLUCOSE result: 130
Tagged concepts can be augmented with relevant information such as lab trends, drug-disease indications, snippets from prior medical records, etc.
HPI: ... h/o dm, htn
PMH: dm, htn
Reduce redundancy in documentation by pre-filling forms and other sections with already-captured information (e.g. information in History of Present Illness sections can be added to Past Medical History).

Figure 1: Semi-structuring notes with contextual autocompletion can enable extensive changes to both documentation and clinical decision support.

able contextual information retrieval about disease history and lab trends, without ever leaving the note interface. Finally, 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 universal vocabulary (the Unified Medical Language System, or UMLS), notes written using our system are semi-structured and parseable.

Generalizable Insights about Machine Learning in the Context of Healthcare 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 (Luo et al., 2019). Even for a human expert, 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 (Chen et al., 2008). In addition, this work demonstrates the power of combining varying noisy and possibly incomplete sources of a patient's medical record to create a context for the current clinical setting that allows us to accurately predict concepts relevant to a new medical event.

2. Background: The Current State of Clinical Workflows

When a patient enters the ED, there are several phases of documentation. First, a triage nurse records patient vitals and a short description of the visit reason. This triage note is then summarized in a succinct phrase known as the Chief Complaint. Doctors also maintain a clinical note which is updated throughout the course of the visit and contains information about the patient's history, current presentation, pertinent labs and tests, and a final diagnosis and treatment plan. This note is also a constantly-evolving document. It 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 note is time-intensive to create, and as such, our work focuses on decreasing documentation burden within the doctor's note.

Clinical staff also have access to the patient's past EHR, which is a rich data source. The bulk of information in EHRs lies within unstructured clinical notes in the patient's file, which contain detailed information about disease history and prior clinical care. Yet these notes are long and difficult to quickly parse– in our dataset, the median number of EHR notes per person is 34 with a median note length of 301 words. There have been attempts to mitigate this information overload by creating semi-structured representations of a patient's medical history such as the problem list, which catalogs a patient's prior conditions. However, these lists are poorly maintained and inconsistent amongst practitioners (Van Vleck et al., 2008).

Efforts to intelligently structure free-text within clinical notes have been limited. One common technique is to pre-fill notes with templates that rely on structured text (Weis and Levy, 2014)– for example, clinical notes usually begin with a summary of patient demographics and the chief complaint like `26 y/o M complains of dyspnea`. This method works for routine cases and structured, repetitive phrases that occur in some sections of notes, but fails to capture subtleties of documentation that reflect the nuances of clinical reasoning and physician preference.

To date, the largest-scale attempt to ease clinical documentation burden with machine learning is by Greenbaum et al. (2017), who built an autocomplete model to predict candidate chief complaints in the ED from a set of approximately 200 standardized options. The model –a multiclass SVM trained on triage information– was used to structure 99% of chief complaints in a live setting. We build on the work in Greenbaum et al. to provide contextual autocompletion functionality for an unstructured clinical note by architecting a model that incorporates both contemporaneous clinical information (triage text, vital signs, and laboratory results) and past medical history (EHR), and by building an interface that supports intuitive and on-the-fly documentation of multiple tagged terms from a large set of clinical concepts.

3. Methods

3.1. Data Overview and Cohort Definitions

We use data from 273,000 anonymized visits to the Beth Israel Deaconess Medical Center (BIDMC) ED over the last decade, representing around 140,000 unique patients. For each

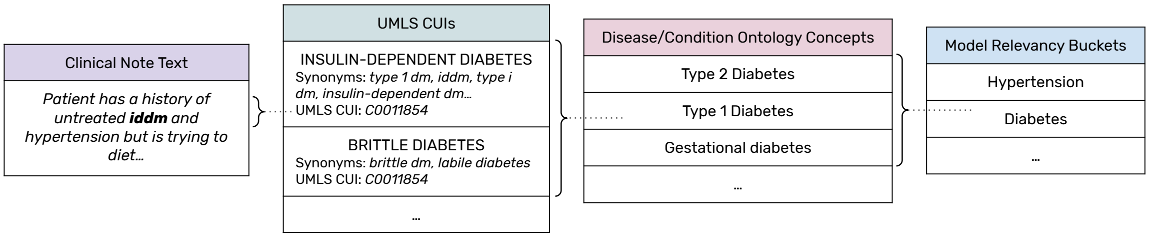

Clinical Note Text
*Patient has a history of untreated **iddm** and hypertension but is trying to diet…*
UMLS CUIs
INSULIN-DEPENDENT DIABETES
Synonyms: *type 1 dm, iddm, type i dm, insulin-dependent dm…*
UMLS CUI: *C0011854*
BRITTLE DIABETES
Synonyms: *brittle dm, labile diabetes*
UMLS CUI: *C0011854*
…
Disease/Condition Ontology Concepts
Type 2 Diabetes
Type 1 Diabetes
Gestational diabetes
…
Model Relevancy Buckets
Hypertension
Diabetes
…

Figure 2: Data model for our ontology of conditions. 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.

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 medications currently prescribed to the patient. In addition, we have all prior EHR notes for patients who had previously been in the BIDMC system (74% of visits). We do not restrict our analysis to any particular subset of these visits.

3.2. Defining Labels for Autocompletion

The goal of autocompletion 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. We then normalize these concepts to UMLS. An example of this is shown in Figure 2.

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 time linear in the note length. We also apply a modified NegEx-style negation detection algorithm as in Chapman et al. (2001) to identify and mark which of these extracted terms occurred within a negative context. We provide additional details on our implementation of negative context detection in Appendix A.2. 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. 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 between closely linked conditions is difficult. 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 improve sample efficiency, we share weights between similar conditions 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 2. The complete revised ontology (consisting of 940 entries encompassing 8,451 unique UMLS CUIs) can be found in Section B.4 of the Appendix.

Condition concepts also represent varying levels of granularity, which is necessary in clinical text (Tange et al., 1997)– 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. 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 2. We find our UMLS-based extraction is equally effective on our text as out-of-the-box learned extraction models such as cTakes (Savova et al., 2010), scispaCy (Neumann et al., 2019), and DistilBERT (Sanh et al., 2019), while being the only paradigm suitably fast for live deployment. We elaborate on this in Section A.3 of the Appendix.

3.3. Developing Predictive Features for Autocompletion

In a typical 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. These representations are state-of-the-art for clinical language modelling (Huang et al., 2019; Wu et al., 2020; Liu, 2018). In our framework, complex inference techniques are too slow to surface live suggestions with low latency in a hospital setting, and we only seek to predict clinical concepts rather than the general language a clinician types.

All features we use as part of our context must be available *before* a patient and physician interact, as this allows us to surface live suggestions as the clinician creates documentation. This limits the data we can incorporate into our autocompletion models to unstructured textual data from the EHR, which is our glimpse into the patient's medical history; and triage-time information such as vitals, chief complaint, and the triage assessment.

3.3.1. 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 3.2. 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. 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.

3.3.2. Featurizing Triage Vitals

Triage vitals are already structured as they represent information that is inherently quantitative, such as heart rate and blood pressure. We discuss specific strategies of further preprocessing triage vitals with each model use-case below.

3.4. Autocompletion Models

We frame contextual autocompletion 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 EHR notes, and the patient's triage vitals. Calling an inference step of our model each time a word is written or removed is prohibitive in terms of latency, so we employ a rules-based approach to incorporate $w_{1:i-1}$ into our prediction while learning how to use the nuanced information in $T, \mathcal{H}, V$. Inference thus only needs to be run once per patient. We first use $w_{1:i-1}$ to determine when the clinician wants to enter a potential clinical concept, and if so, whether that concept is a condition, symptom, lab, or medication. We then 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 he/she 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 four concept-specific ranking models:

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 note 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.

3.4.1. 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 EHR 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 5 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 $\mathbb{1}[b \in \mathcal{H}]$ of each model relevancy bucket b in prior EHR notes $\mathcal{H}$. Each arm of the network is passed through a single dense layer with rectified linear unit activation, the two outputs of the dense layers concatenated, and then the combined embedding is passed into a final dense layer with sigmoid activation to provide a vector of estimates of relevancy for each bucket. We recover a term-wise ranking by sorting each term first by whether it appears in the EHR, 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. Training details for this architecture can be found in Section B.2. 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 $P(b|T, \mathbb{1}[b \in \mathcal{H}])$. 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 $P(\mathbb{1}[b \in \mathcal{H}])$ 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 $P(b|T, D, \mathbb{1}[b \in \mathcal{H}])$ 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.

3.4.2. Autocompleting Present 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 can be found in the Section A.4. 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.

3.4.3. 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.

3.4.4. Determining Autocompletion Scope and Type

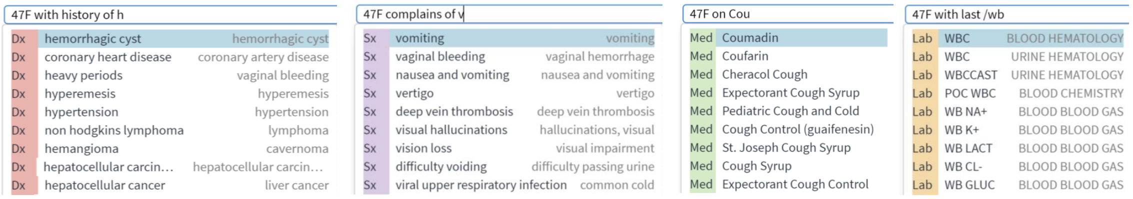
There are two components to displaying autocompletion suggestions: (1) the *scope* of autocompletion, which determines when a clinician wants to document a concept; and (2) the *type* of autocompletion, which determines a ranking over whether the clinician wants to document a condition, symptom, lab, or medication. A potential approach to this problem is to build a sequential model predicting whether the next word typed will be a clinical concept and its corresponding type, but this requires significant client-side infrastructure to curb model latency– Gmail's Smart Compose system, for example, which surfaces dynamic suggestions of words to type from a neural language model, is only made possible via custom hardware and extensive system infrastructure (Chen et al., 2019). We discuss this further in Section 5. To build a system which can run live, we instead adopt a 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 autocompletion 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 autocompletion type and scope by greedily matching triggers in the text (Chapman et al., 2001). A full algorithm sketch of this is included in the Section A.2, and screenshots of the scope and type prediction algorithms at work are shown in Figure 3.

While we rely on autocompletion scope and type prediction algorithms to guess where the user will insert a tagged term and the types of these terms, we support fallback data capture methods for when our algorithms fail. We do so in two ways. 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 (Aho and Corasick, 1975). 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 10 of the Appendix. We analyze how often these mechanisms are exercised in practice below. Our rule-based algorithms have an average end-to-end latency of ≈ 0.2 milliseconds to make a prediction, which is well below the 100ms threshold for a response to feel instantaneous Nielsen (1993). In contrast, making an API call to a shallow convolutional neural network for scope and type prediction takes upwards of 250ms in the absence CPU throttling, network overload, etc.

4. Results

A physician uses contextual autocompletion by naturally typing a note and either automatically or retroactively completing clinical phrases that are then rendered as tagged concepts.



47F with history of h

Dx	hemorrhagic cyst	hemorrhagic cyst
Dx	coronary heart disease	coronary artery disease
Dx	heavy periods	vaginal bleeding
Dx	hyperemesis	hyperemesis
Dx	hypertension	hypertension
Dx	non hodgkins lymphoma	lymphoma
Dx	hemangioma	cavernoma
Dx	hepatocellular carcin…	hepatocellular carcin…
Dx	hepatocellular cancer	liver cancer

47F complains of v

Sx	vomiting	vomiting
Sx	vaginal bleeding	vaginal hemorrhage
Sx	nausea and vomiting	nausea and vomiting
Sx	vertigo	vertigo
Sx	deep vein thrombosis	deep vein thrombosis
Sx	visual hallucinations	hallucinations, visual
Sx	vision loss	visual impairment
Sx	difficulty voiding	difficulty passing urine
Sx	viral upper respiratory infection	common cold

47F on Cou

Med	Coumadin
Med	Coufarin
Med	Cheracol Cough
Med	Expectorant Cough Syrup
Med	Pediatric Cough and Cold
Med	Cough Control (guaifenesin)
Med	St. Joseph Cough Syrup
Med	Cough Syrup
Med	Expectorant Cough Control

47F with last /wb

Lab	WBC	BLOOD HEMATOLOGY
Lab	WBC	URINE HEMATOLOGY
Lab	WBCCAST	URINE HEMATOLOGY
Lab	POC WBC	BLOOD CHEMISTRY
Lab	WB NA+	BLOOD BLOOD GAS
Lab	WB K+	BLOOD BLOOD GAS
Lab	WB LACT	BLOOD BLOOD GAS
Lab	WB CL-	BLOOD BLOOD GAS
Lab	WB GLUC	BLOOD BLOOD GAS

Figure 3: 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.

As in standard autocomplete, autocompleted concepts are filtered to those that match the user's typed prefix. We briefly describe the user experience of the tool with a screenshot in Figure 3, and examine how it reduces clinical documentation burden in practice.

4.1. Performance and Usability Metrics

4.1.1. 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. We use two standard information retrieval metrics, the *mean reciprocal rank* (MRR) and *mean average precision* (MAP) to gauge the quality of our rankings (definitions in Section B.3). We compare the MRR of our contextual autocompletion tool rankings against two naive baselines: spell-based autocompletion (ranking terms alphabetically) and frequency-based autocompletion (ranking terms by frequency). Performance by MAP is detailed in the Section B.3.

To generate our evaluation set, we extract medical concepts from 25,000 clinical notes with the technique outlined in Section 3.3.1. Using the order in which concepts were suggested, we first measure MRR assuming our scope and type detection was perfect, broken down by the four concept types. Results are shown in Figure 4. We see the largest gain in using a contextual model for conditions, because the space of terms is large and the richness of the EHR 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 who did not have any history on file at the hospital. 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 autocompletion scope and type detection algorithm, we also measure MRR when typing HPI sections of notes. We focus on HPI notes as they contain a range of concept types (conditions, symptoms, medications, etc.) and also were the only note section we could reliably segment due to dataset limitations. On average, there are 6.8 documented clinical concepts per HPI section. Of the extracted clinical concepts in HPI sections, 46% of terms were autocompleted automati-

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

Figure 4: Retrospective Evaluation of MRR using Contextual Autocompletion. We report average MRR (±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.

cally without a manual trigger, and in 77% of those cases, we guessed autocompletion 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 Figure 4. These manually-prompted scenarios can be mitigated as a doctor learns and adapts to the autocompletion triggers of the system, which we elaborate on in Section 5.

4.1.2. Documentation in the Wild: Live Evaluation

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

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

Figure 5: Live Evaluation of Contextual Autocompletion 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.

behavior– there may be a delay between a term being suggested first and when a clinician actually autocompletes the term. We compare keystroke burden between a contextual model and no autocompletion in Figure 5. 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. Moreover, 96% of these terms were automatically prompted (as opposed to the user manunally prompting the autocomplete), indicating our scope and type detection had high recall even when doctors had not yet adapted to the system. For 77% of the terms tagged without a retroactive label, we also predicted the clinical concept type correctly.

4.2. Autocompletion Sensitivity Analysis

Concept frequency influences the efficacy of our contextual autocompletion 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 Figure 6, 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

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

Figure 6: 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.

largely be attributed to the inherent bias of our ranking scheme, which preferentially orders terms mentioned in the EHR above those that are not.

4.3. Interpreting Contextual 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.

4.3.1. Performance By Concept

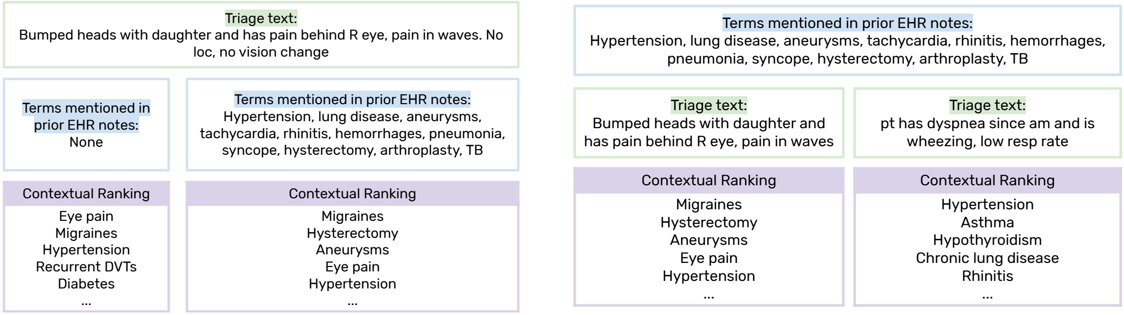
Our multi-label model predicts the binary relevance of each model relevancy bucket. To better interpret relevancy predictions on a per-bucket level, we approximate our model for a specific relevancy bucket b with a linear function of the inputs. This is done by fitting a L_1-regularized linear approximation between the features and the logits generated by the model for bucket b to surface highly-weighted features (Liu et al., 2018). In Figure 7, 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 12 of the Appendix. Interestingly, while all of the chosen concepts 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.

4.3.2. Qualitative Evaluation & Readability

We qualitatively evaluate rankings over conditions to better understand model decisions. As can be seen in the selected examples in Figure 8, both the presence of EHR 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 8a, two patients have identical triage text but different

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

Figure 7: 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 3.2.


Triage text:
Bumped heads with daughter and has pain behind R eye, pain in waves. No loc, no vision change
Terms mentioned in prior EHR notes:
None
Terms mentioned in prior EHR notes:
Hypertension, lung disease, aneurysms, tachycardia, rhinitis, hemorrhages, pneumonia, syncope, hysterectomy, arthroplasty, TB
Contextual Ranking
Eye pain
Migraines
Hypertension
Recurrent DVTs
Diabetes
...
Contextual Ranking
Migraines
Hysterectomy
Aneurysms
Eye pain
Hypertension
...
Terms mentioned in prior EHR notes:
Hypertension, lung disease, aneurysms, tachycardia, rhinitis, hemorrhages, pneumonia, syncope, hysterectomy, arthroplasty, TB
Triage text:
Bumped heads with daughter and has pain behind R eye, pain in waves
Triage text:
pt has dyspnea since am and is wheezing, low resp rate
Contextual Ranking
Migraines
Hysterectomy
Aneurysms
Eye pain
Hypertension
...
Contextual Ranking
Hypertension
Asthma
Hypothyroidism
Chronic lung disease
Rhinitis
...

(a) Effect of patient history on contextual rankings.

(b) Effect of triage note on contextual rankings.

Figure 8: Case Studies of Rankings over Conditions

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 8b, two patients with identical medical histories but differing chief complaints have vastly different context-specific rankings.

5. Discussion

The contextual autocompletion tool 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. The ablation tests we carried out show that using a few features (primarily representations of medical histories) can result in performant predictive 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.

Our system provides automatic and natural autocompletion of a clinical concept when our scope detection algorithm is accurate, and users may need to resort to a manual trigger in other cases. This is not a major hindrance based on our evaluation criteria, and we significantly reduce documentation burden even if the manual trigger is used. However, we note that more complex model classes (e.g. recurrent neural networks/sequential learning models) may be better at scope prediction. These models can introduce significant client-side latency because they require performing inference after each character-level change of the note. Thus, to allow for straightforward integration into the existing hospital ecosystem, we do not explore these schemes in this paper. Future iterations of our tool might automatically learn a set of autocompletion triggers rather than using hand-crafted rules. Using manual triggers for autocompletion, however, can also establish consistent system behavior for physicians, and create notes that are concise. 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`*.

We propose two other future directions to build on and continue this work. The first is to better integrate key semantic modifiers into our tool. As an example, doctors often document the absence of symptoms (e.g. `no fever`) to aid in a differential diagnosis. While we can use rule-based approaches to retrospectively attach negation modifiers to tagged medical concepts, future work should seek to fuse modifier capture with the UI. Clinicians might also type a term that refers to someone other than the patient (e.g. `family history of diabetes in mother`), and we should automatically learn that the concept `diabetes` refers to a third-party rather than the patient.

In addition to facilitating semantic modifier capture, a next iteration should dynamically update suggested terms to document using already-tagged terms in the note. Tagging `atrial fibrillation`, 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 in this first iteration.

6. 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. While our contextual autocompletion tool can reduce documentation burden and curate clean data for machine learning purposes, it also opens the possibility of reforming clinical documentation practices to make notes more understandable to humans and algorithms alike. Fundamentally, live-tagging of medical concepts enables unprecedented changes to EHR design. By integrating machine learning methodologies into documentation practices, we can usher in a new era of EHRs that assist rather than impede physicians.

Acknowledgments

We thank the MIT Abdul Latif Jameel Clinic for Machine Learning in Health for the grant that funded this work.

References

A. V. Aho and M.J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975. ISSN 0001-0782. doi: 10.1145/360825.360855. URL https://doi.org/10.1145/360825.360855.

Ilseyar Alimova and Elena Tutubalina. Multiple features for clinical relation extraction: a machine learning approach. *Journal of Biomedical Informatics*, 103:103382, 02 2020. doi: 10.1016/j.jbi.2020.103382.

D. Aljabri, A. Dumitrascu, C. Burton, L. White, M. Khan, S. Xirasagar, R. Horner, and J. 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. ISSN 1472-6947. doi: 10.1186/s12911-018-0644-4. URL https://doi.org/10.1186/s12911-018-0644-4.

P. Carayon, T. Wetterneck, B. Alyousef, R. Brown, R. Cartmill, K. McGuire, P.L.T. Hoonakker, J. Slagle, K. Roy, J. Walker, M. Weinger, A. Xie, and K. 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. doi: 10.1016/j.ijmedinf.2015.04.002.

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

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. ISBN 9781450362016. doi: 10.1145/3292500.3330723. URL https://doi.org/10.1145/3292500.3330723.

Y. Chen, H. Gu, Y. Perl, and J. Geller. Structural group-based auditing of missing hierarchical relationships in umls. *Journal of biomedical informatics*, 42:452–67, 09 2008. doi: 10.1016/j.jbi.2008.08.006.

M. A. Cretikos, R. Bellomo, K. Hillman, J. Chen, S. Finfer, and A. Flabouris. Respiratory rate: the neglected vital sign. *Med. J. Aust.*, 188(11):657–659, Jun 2008.

J. Devlin, M. Chang, K. Lee, and K. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

R. L. Gardner, E. Cooper, J. Haskell, D. A. Harris, S. Poplau, P. J. Kroth, and M. Linzer. Physician stress and burnout: the impact of health information technology. *Journal of American Medical Informatics Association*, 26(2):106–114, Feb 2019.

M. Gerard, H. Chimowitz, A. Fossa, F. Bourgeois, L Fernandez, and SK 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. ISSN 1438-8871. doi: 10.2196/jmir.9196. URL http://www.jmir.org/2018/5/e191/.

N.R. Greenbaum, Y. Jernite, Y. Halpern, S. Calder, L.A. Nathanson, D. Sontag, and S. 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. doi: 10.1101/127092. URL https://www.biorxiv.org/content/early/2017/04/12/127092.

K. Huang, J. Altosaar, and R. Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *CoRR*, abs/1904.05342, 2019. URL http://arxiv.org/abs/1904.05342.

A.A. Khorana. Physician as typist. *Journal of Clinical Oncology*, 28(24):3899–3900, 2010. doi: 10.1200/JCO.2010.29.4504. URL https://doi.org/10.1200/JCO.2010.29.4504. PMID: 20547988.

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. ISSN 1369-7625. doi: 10.1111/j.1369-7625.2009.00555.x. URL https://pubmed.ncbi.nlm.nih.gov/19709316. 19709316[pmid].

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. doi: 10.18653/v1/N16-1030. URL https://www.aclweb.org/anthology/N16-1030.

Peter J. Liu. Learning to write notes in electronic health records. *CoRR*, abs/1808.02622, 2018. URL http://arxiv.org/abs/1808.02622.

Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. *CoRR*, abs/1812.10924, 2018. URL http://arxiv.org/abs/1812.10924.

Y. Luo, W. Sun, and A. Rumshisky. Mcn: A comprehensive corpus for medical concept normalization. *Journal of Biomedical Informatics*, 92:103132, 02 2019. doi: 10.1016/j.jbi.2019.103132.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. Scispacy: Fast and robust models for biomedical natural language processing. *CoRR*, abs/1902.07669, 2019. URL http://arxiv.org/abs/1902.07669.

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

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.

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.

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.

Fred Schulte and Erika Fry. Death by 1,000 clicks: Where electronic health records went wrong. *Kaiser Health News*, Jun 2019. URL https://khn.org/news/death-by-a-thousand-clicks/.

C. Smith, S. Hetzel, P. Dalrymple, and A. Keselman. Beyond readability: Investigating coherence of clinical text for consumers. *Journal of medical Internet research*, 13:e104, 10 2011. doi: 10.2196/jmir.1842.

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.

T. T. Van Vleck, A. Wilcox, P. D. Stetson, S. B. Johnson, and N. Elhadad. Content and structure of clinical problem lists: a corpus analysis. *AMIA Annual Symposium Proceedings*, pages 753–757, Nov 2008.

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

S. Wu, K. Roberts, S. Datta, J. Du, Z. Ji, Y. Si, S. Soni, Q. Wang, Q. Wei, Y. Xiang, B. Zhao, and H. Xu. Deep learning in clinical natural language processing: a methodical review. *Journal of American Medical Informatics Association*, 27(3):457–470, Mar 2020.

Appendix A. Data Extraction and Featurization

A.1. 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 synthetic but mimic the formatting and style of real data.

Triage Note

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

Chief Complaint

```
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.2. The NegEx Algorithm

We use a version of the NegEx algorithm Chapman et al. (2001) 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 in Figure 9.

A.3. Trie-Based Extraction of UMLS Concepts

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 hepatitic 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:

```
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 9: Pseudocode of the rule-based negation detection algorithm.

- `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 (Savova et al., 2010). cTakes is an older system that often misses clinical abbreviations (Rojas and Ratté, 2018). 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` (Neumann et al., 2019). 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. (2016). `scispaCy` identifies clinical and biomedical terms on the text first with its entity recognition model, and then retroactively maps this to UMLS using a string match over synonyms.

- BERT-based clinical entity extraction models such as Alimova and Tutubalina (2020), which combine a transformer 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 Alimova and Tutubalina (2020) due to the lack of labelled 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 (Sanh et al., 2019), and train on a custom

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 diambiguated, 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 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.

vocabulary which is smaller than that of the original BERT model (Devlin et al., 2019).

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 labelled 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.

A.4. Bucketization of Triage Vitals

As described in Section 3.4.2, 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 (Cretikos et al., 2008). 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` (e.g. below 18), `18-33`, `34-48`, `48-64`, `64-77`, and `78+`.

Appendix B. Extended Autocomplete Performance

B.1. Autocompletion Scope and Type Detection

Here, we provide an algorithm sketch of our autocompletion scope and type detection framework. The algorithm greedily uses keywords that act as autocompletion triggers, and is run and updated as a physician types a clinical note. First, we initialize the scope and type of our autocompletion to be null. Then, for each word w in the text, we update the scope accordingly:

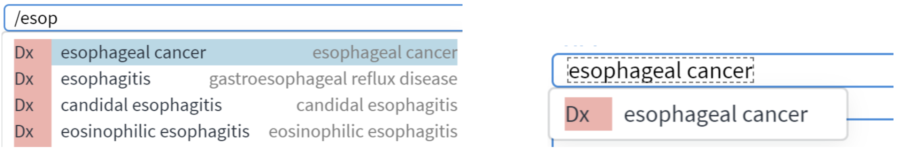
- If w is part of a autocompletion trigger phrase such as `presents with`, we turn the autocompletion scope on and suggest terms to the user. We set the autocompletion type based on the trigger (`presents with` maps to `SYMPTOM`.)
- If w is a continuation token such as `and`, `or`, or `,`, we maintain 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.

B.2. Training Contextual Model for Conditions

Our contextual model to predict a ranking over conditions is a dual-branched network that 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 3.2) in the patient's prior medical history. This is a length-227 binary vector.


/esop
Dx esophageal cancer esophageal cancer
Dx esophagitis gastroesophageal reflux disease
Dx candidal esophagitis candidal esophagitis
Dx eosinophilic esophagitis eosinophilic esophagitis
esophageal cancer
Dx esophageal cancer

(a) Manual autocompletion trigger

(b) Retroactive tagging

Figure 10: Screenshots of our backup data capture strategies in the case that autocompletion scope detection algorithms fail. (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.

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.

B.3. Retrospective Autocompletion Performance using MRR, MAP, and Keystroke Burden

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 $R = \{r_1, r_2, r_3, \cdots\}$ of suggested terms and a ground truth set of terms that the clinician wants to documented denoted by $T = \{r_{\pi(1)}, r_{\pi(2)}, r_{\pi(3)}, \cdots\}$. We define the MRR of these suggestions as

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

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:

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

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

Below, we compare the various models we prototyped to predict each clinical concept type with MAP and keystroke burden. Results in terms of MRR are in Figure 4.

B.4. Ontologies and Code

The codebase for our analyses as well as our publicly-available ontologies for conditions, symptoms, labs, and medications can be found here: https://github.com/clinicalml/ContextualAutocomplete_MLHC2020.

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

Figure 11: 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

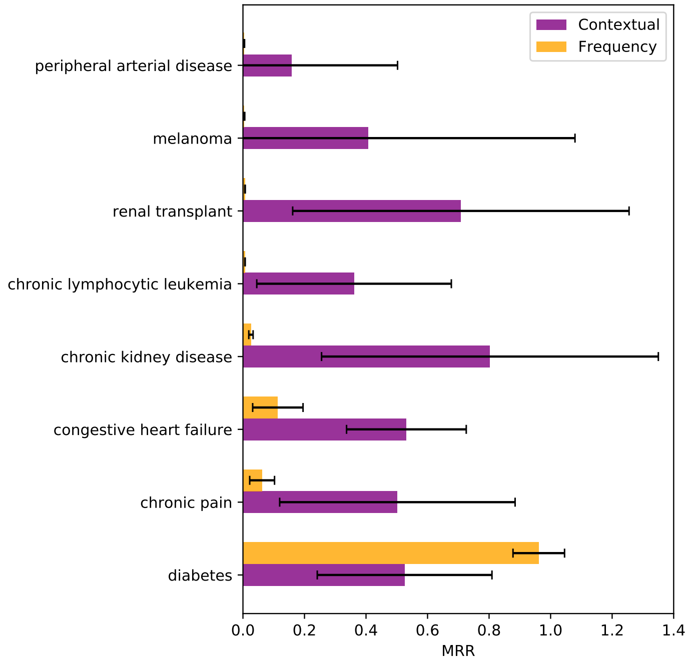


Figure 12: Mean MRR for five conditions (± 95% CI from mean) using contextual and frequency-based autocompletion. Concepts were chosen to get representative samples of the data.