

Natural Language Processing for Precision Clinical Diagnostics and Treatment

by

Isabel Chien

S.B., C.S. Massachusetts Institute of Technology (2017)

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

June 2018

© Isabel Chien. 2018. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and to
distribute publicly paper and electronic copies of this thesis document
in whole or in part in any medium now known or hereafter created.

Author

Department of Electrical Engineering and Computer Science

May 25, 2018

Certified by

Manolis Kellis

Professor

Thesis Supervisor

Accepted by

Katrina LaCurts

Master of Engineering Thesis Committee

Natural Language Processing for Precision Clinical Diagnostics and Treatment

by

Isabel Chien

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, I focus upon application of natural language processing to clinical diagnostics and treatment within the palliative care and serious illness field. I explore a variety of natural language processing methods, including deep learning, rule-based, and classic machine learning, and applied to the identification of documentation reflecting advanced care planning measures, serious illnesses, and serious illness symptoms. I introduce two tools that can be used to analyze clinical notes from electronic health records: ClinicalRegex, a regular expression interface, and PyCCI, an a clinical text annotation tool. Additionally, I discuss a palliative care-focused research project in which I apply machine learning natural language processing methods to identifying clinical documentation in the palliative care and serious illness field. Advance care planning, which includes clarifying and documenting goals of care and preferences for future care, is essential for achieving end-of-life care that is consistent with the preferences of dying patients and their families. Physicians document their communication about these preferences as unstructured free text in clinical notes; as a result, routine assessment of this quality indicator is time consuming and costly. Integrating goals of care conversations and advance care planning into decision-making about palliative surgery have been shown to result in less invasive care near the time of death and improve clinical outcomes for both the patient and surviving family members. Natural language processing methods offer an efficient and scalable way to improve the visibility of documented serious illness conversations within electronic health record data, helping to better quality of care.

Thesis Supervisor: Manolis Kellis
Title: Professor

Acknowledgments

I would like to thank Manolis Kellis for acting as my thesis advisor and welcoming me to the Kellis lab. Special thanks goes to Alvin Shi, who has acted as my mentor for the two years I have spent in the lab and who has helped me immensely throughout my research work.

Thank you to Dr. Charlotta Lindvall, who has acted as the principal investigator for the several projects I have been involved with at Dana-Farber Cancer Institute in my Master's research. I deeply appreciate your constant support and encouragement, and it has motivated me immeasurably in my work.

I am particularly grateful to Tristan Naumann, Alex Chan, Franck Dernoncourt, Elena Sergeeva, Edward Moseley, and Alistair Johnson for helpful guidance and advice during the development of this research. Additionally, I would like to thank Peter Szolovits for providing additional computing resources, as well as Saad Salman, Sarah Kaminar Bourland, Haruki Matsumoto and Dickson Lui for annotating clinical notes.

Contents

1	Introduction and overview	15
2	ClinicalRegex: A Regular Expression Program	19
2.1	Introduction	19
2.2	Architecture	20
2.3	Features and Functionality	21
2.4	Application to identifying seriously ill patients	23
2.4.1	Data	23
2.4.2	Approach	24
2.4.3	Results	25
2.5	Application to assessment of palliative care processes	26
2.5.1	Data source and study population	27
2.5.2	Methods	27
2.5.3	Results	28
3	PyCCI: A Clinical Note Annotation Tool	31
3.1	Introduction	31
3.2	Architecture	31
3.3	Features and Functionality	34
3.3.1	Annotating	36
3.3.2	Reviewing	38
3.3.3	File Formats	38
3.4	Applications	39

4	Natural Language Processing for Advanced Care Planning	43
4.1	Introduction and related work	43
4.2	Data	44
4.2.1	Data Source	44
4.2.2	Cohort	45
4.2.3	Clinical domains	46
4.2.4	Annotation	46
4.3	Methods	47
4.3.1	Pre-processing	47
4.3.2	Regular expression	47
4.3.3	Artificial neural network	48
4.4	Results	50
4.4.1	Evaluation metrics	50
4.4.2	Performance	50
4.4.3	Error analysis	51
4.4.4	Effect of training set size	52
4.5	Conclusions and future work	53
A	Tables	57
A.1	Regular expression library	57
A.2	Token-level performance	57
A.3	Examples of identified text	58

List of Figures

2-1	Software Architecture of ClinicalRegex	20
2-2	ClinicalRegex user interface elements. 1: File selection button. Can accept RPDR text files or any CSV file with note and ID columns. Displays file name here. 2: Output file selection button. The user can elect to view already created output files. When the user runs the regular expression on a selected file, the output file automatically appears here. 3: Clinical text box, where the clinical note is displayed with identified text highlighted in yellow. 4: Patient ID displayed here, or whatever column value is specified in the Patient ID column key. 5: Previous and Next buttons navigate to the previous or next note. 6: Text boxes where you can specific the column name where the clinical note and patient IDs reside. Must be correctly specified if the input file is not an RPDR file. Is hidden when RPDR format is selected. 7: Regular expression functionality occurs here. The user can specify any output file name they would like, the default is “output.csv.” They can enter comma-separated texts into the textbox. These are the keywords the regular expressions will extract. To run the regular expression on the input file, the user presses the “Run Regex” button. 8: Annotation text box. When the regular expression is complete, or if the user opens an output file, they can annotate each note with a value. When they press “Save” this value is written to file.	22
2-3	Three step process of patient identification	24

3-1	Software Architecture of PyCCI	32
3-2	PyCCI typical workflow. Files are in green, actions in orange, PyCCI-use in blue.	37
3-3	PyCCI General user interface elements. 1: File selection buttons. “Open CSV” opens a file for annotating, “Open Results” allows the user to select any number of files for review and comparison of completed annotations. 2: Back and Next buttons allows the user to navigate through notes. 3: Clinical notes text box, displays the current note. 4: Indicators are of the configured annotation categories. The user pastes text they wish to label into the corresponding box. The check box is automatically filled. 5: The “None” checkbox is used to indicate that a note has been reviewed and contains no labelled text. 6: Highlighted text. This is text that has been labelled and saved. The darker blue portion is text that has been labelled in multiple categories. 7: Popup menu on clicked highlighted text. Displays what category the text is labelled as and provides delete commands. 8. Save button. Pressed to save the annotations currently in the indicator text boxes.	41
3-4	PyCCI Symptoms review mode user interface elements. 1: Popup menu on clicked highlighted text. Displays the corresponding text snippet, the labels assigned by annotators, and options to add or delete labels. You can see that labelled texts with agreeing labels from annotators are in green, while conflicting labels are in red. 2: Save button. This button saves the annotations to file.	42
4-1	Pipeline used to train and validate neural networks to identify ACP documentation	49
4-2	Comparison between the F1-score of the regular expression method and neural networks by domain.	53

4-3 Neural network performance on validation set for detection of note-level documentation of patient care preferences by number of notes used for training. 53

List of Tables

2.1	Regular expression keyword library for assessment of palliative care processes	29
2.2	Performance of regular expression for identification of quality measures	30
3.1	PyCCI configuration parameters	34
4.1	Sample characteristics	45
4.2	Clinical domain specifications.	46
4.3	Performance (%) of the regular expression method on the validation data set.	51
4.4	Performance (%) of the neural networks on the validation data set. Values in parentheses are 95% confidence intervals.	52

Chapter 1

Introduction and overview

This thesis focuses upon application of natural language processing to clinical diagnostics and treatment within the palliative care and serious illness field. A variety of natural language processing methods are explored, including deep learning, rule-based, and classic machine learning, and applied to the identification of documentation reflecting advanced care planning measures, serious illnesses, and serious illness symptoms.

To ensure that patients receive care that is consistent with their goals, clinicians must communicate with seriously ill patients about their treatment preferences. More than 80% of Americans say they would prefer to die at home, if possible. Despite this, 60% of Americans die in acute care hospitals and 20% die in an Intensive Care Unit (ICU)[8]. Advance care planning, which includes clarifying and documenting goals of care and preferences for future care, is essential for achieving end-of-life care that is consistent with the preferences of seriously ill patients and their families. Inadequate communication is associated with more aggressive care near the the time of death, decreased use of hospice and increased anxiety and depression in surviving family members[51, 41, 47, 13]. Several studies have demonstrated the potential of advanced care planning to improve end-of-life outcomes (e.g., reducing unintended ICU admissions and increasing hospice enrollment). In the absence of explicit goals of care decisions, clinicians may provide clinical care[21] that does not provide a meaningful benefit to the patient[22] and, in the worse case, interferes with the treatment of other

patients[21]. For these reasons, it is recommended that care preferences are discussed and documented in the EHR within the first 48 hours of an ICU admission[16, 25].

In recent years a consensus has emerged that such conversations are an essential component of practice and must be monitored to improve care quality. However, the difficulty of retrieving documentation about these conversations from the electronic health record has limited rigorous research on the prevalence and quality of clinical communication. For example, the National Quality Forum (NQF) recommends that goals of care be discussed and documented in the EHR within the first 48 hours of an ICU admission, especially among frail and seriously ill patients. This was one of only two Centers for Medicare and Medicaid Services recommended palliative care quality measures for the Medicare Hospital Inpatient Quality Reporting program[43]. Yet, despite widespread support, routine assessment of this and similar quality measures have proven nearly impossible because the information is embedded as non-discrete free-text within clinical notes. Manual chart review is time-consuming and expensive to scale [50, 14, 1]. Consequently, many end-of-life quality metrics are simply not assessed, and their impact on distal and important patient outcomes have been insufficiently evaluated.

Additionally, palliative surgical procedures, which are performed to reduce symptoms or improve quality of life, are common in patients with advanced cancer. Palliative surgery has been shown to represent 6-20% of all operations performed by surgical oncologists and over 1,000 procedures per year at tertiary cancer centers[36, 26, 39], and national trends point to an increase in the proportion of patients with cancer who receive palliative surgery in their last months or weeks of life[28]. Despite considerable achievements, including contributions from the National Quality Forum (NQF) Palliative and End-of-Life Care Project, healthcare measurement for surgical palliation remains an important gap area.

There is a lack of high-quality evidence or consensus surrounding the appropriate application of palliative surgery. While the risks of mortality and major complications have been described in the literature, there is a paucity of evidence regarding other important patient outcomes after palliative surgery, including symptom relief or reduced

treatment burdens (e.g., prolonged hospitalization). Integrating goals of care conversations and advance care planning into decision-making about palliative surgery have been shown to result in less invasive care near the time of death and improve clinical outcomes for both the patient and surviving family members[51, 41, 47, 13, 38]. However, the extent to which these discussions are implemented and documented has not been quantified.

Over the past twenty years, health services research has undergone a period of explosive growth[45]. These gains have been made possible through improvements in computer processing and accessibility to high-quality national databases. However, the field remains dependent on extraction specialists to facilitate analysis. Trained nurses sift through charts isolating relevant information, which is then compiled and uploaded into the appropriate database[6, 17]. Use of structured data such as administrative claims codes can obviate this process, but at the cost of a reduction in granularity and specificity. The vast majority of relevant patient information (70-80%), resides in unstructured free-text notes[40]. With the widespread adoption of electronic health records (EHR), it is possible to analyze clinical notes using powerful computational methods such as natural language processing (NLP) and machine learning[27, 5]. These methods are particularly relevant in fields such as palliative medicine, where standard administrative data poorly captures the relevant patients population[24].

With the increasing amount of medical data becoming available, as well as computational techniques becoming more widespread, it is important for clinicians to be empowered to take on computationally-driven research. Clinicians come into contact with huge magnitudes of data on a daily basis, yet much of it is indecipherably complex. Methods to process large amounts of data are often inaccessible to the typical clinician and require more advanced computational knowledge. With easy-to-use software tools, clinicians could begin to contribute more readily to data-driven computational research and also perform some of that research themselves.

In this thesis I introduce two tools, which I have developed, that can be used to analyze clinical notes from electronic health records. Additionally, I discuss a

palliative care-focused research project in which I apply machine learning natural language processing methods to identifying clinical documentation in the palliative care and serious illness field.

Chapter 2 discusses ClinicalRegex, a natural language processing interface that allows the user to easily run regular expression on clinical notes. Chapter 3 discusses PyCCI, a clinical notes annotation tool. There are two versions: one for symptoms annotation, and one for general annotations. PyCCI is designed to be used by clinicians without the need of technical intervention. Chapter 4 discusses a study in which I trained and validated a deep neural network, in comparison with rule-based methods, to detect documentation of advanced care planning conversations in clinical notes from electronic health records.

Chapter 2

ClinicalRegex: A Regular Expression Program

2.1 Introduction

Unstructured clinical notes represent 70-80% of all data in electronic health records. They are typically embedded in such data, and manual review of such notes take intensive time and labor [47]. However, important data is still captured in these notes, though buried by volume and lack of organization. With the use of natural language processing, clinical notes can be rapidly scanned to detect pre-specified indicators[40].

Regular expressions are a simple but powerful method of extracting information from text. Many entity extraction tasks can be fulfilled through the usage of a well-validated set of regular expression rules [30]. Regular expressions identify patterns of characters exactly as they are specified in a set of rules. Often, regular expressions are used during initial forays into natural language processing. They are simple, easy-to-understand, and behave exactly as specified. Not only that, but they are still effective in identifying a variety of defined concepts. However, the creation of regular expressions involves a non-trivial amount of technical computer science knowledge and are often inaccessible to most clinicians.

ClinicalRegex is a user interface designed to be easily accessible and intuitive to

navigate. It is able to process regular expressions on input text, with user-specified keywords. With this interface, clinicians can determine keywords to extract and easily perform regular expression extractions with those keywords, without additional technical intervention. In this chapter, I discuss the architecture, features, and functionality of ClinicalRegex, and also describe two research projects in which this software was used. It has been used to identify seriously ill patients with understudied disease processes as well as to assess end-of-life quality indicators in cancer patients receiving palliative surgery.

2.2 Architecture

ClinicalRegex is implemented using Python 3.5 and leverages Tkinter, a graphical user interface Python package. It is implemented using a Model View Controller architecture, which is common to graphical user interface programs. Figure 2-1 displays architecture of this program. The program is compiled into an executable file using PyInstaller, which allows it to be easily utilized by someone unfamiliar with computer programming.

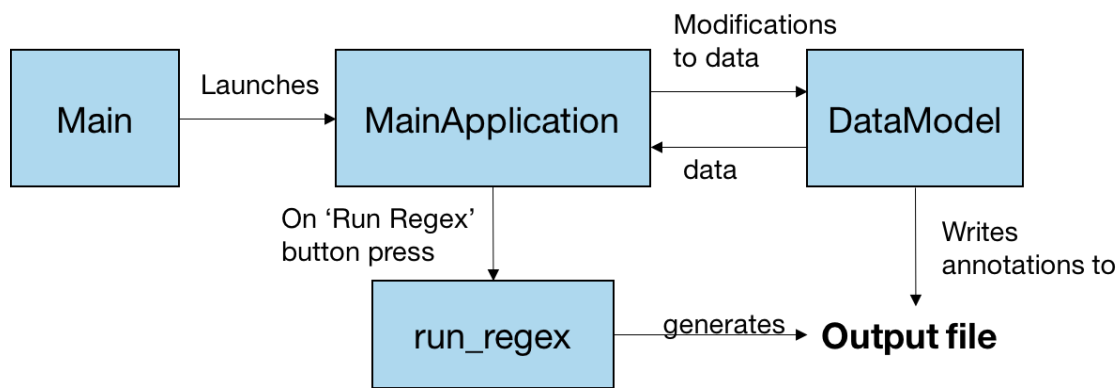


Figure 2-1: Software Architecture of ClinicalRegex

The **MainApplication** class is the “view” class, which is responsible for defining the user interface of ClinicalRegex. Because this is a Tkinter application, the entire program is written in Python. The entire user interface is defined in Python. Due to this, the **MainApplication** class also acts as the “controller” class, which manipulates

the user interface and determines what the display will be. These actions include file selection, user options, and display of the clinical notes. Additionally, a separate file, `extract_values.py`, contains the classes and logic responsible for the regular expression functionality. This code is kept separate for modularity and ease of testing. Finally, the `DataModel` class acts as the “model” class, where data, including file names, dataframes, and indices are stored and modified. `ClinicalRegex` allows annotation of the notes displayed; the logic to write these annotations to file is also included in the `DataModel` class.

2.3 Features and Functionality

`ClinicalRegex` can be used entirely through the user interface. Figure 2-2 displays the different elements of the user interface in detail. In order to perform the regular expression information extraction, a file must be selected by the user via a button press and file selection window. It is compatible with RPDR (Research Patient Data Registry), the file format used by Partners HealthCare to deliver data. It also accepts custom files in CSV formats, needing only a note column key and a patient ID column key so that it can recognize the columns that contain the note text and an ID respectively.

There is a text box on the right hand panel where a list of comma-separated keywords can be typed. These are the keywords that the regular expressions identify within the notes. The regular expression rules take into consideration possible variations in punctuation and text cases, which them more effective than a simple keyword search. The text box next to the “Run Regex” button, which performs the regular expression extraction, allows for modification of the output file name. The output file is saved in the same directory as the original file selected.

This software also allows for human review of the extracted keywords. As soon as the regular expression information extraction is complete, the notes are displayed in the left hand panel. Users are able to look through all notes in the original file and see the relevant keywords highlighted in yellow. A checkbox above the text display panel

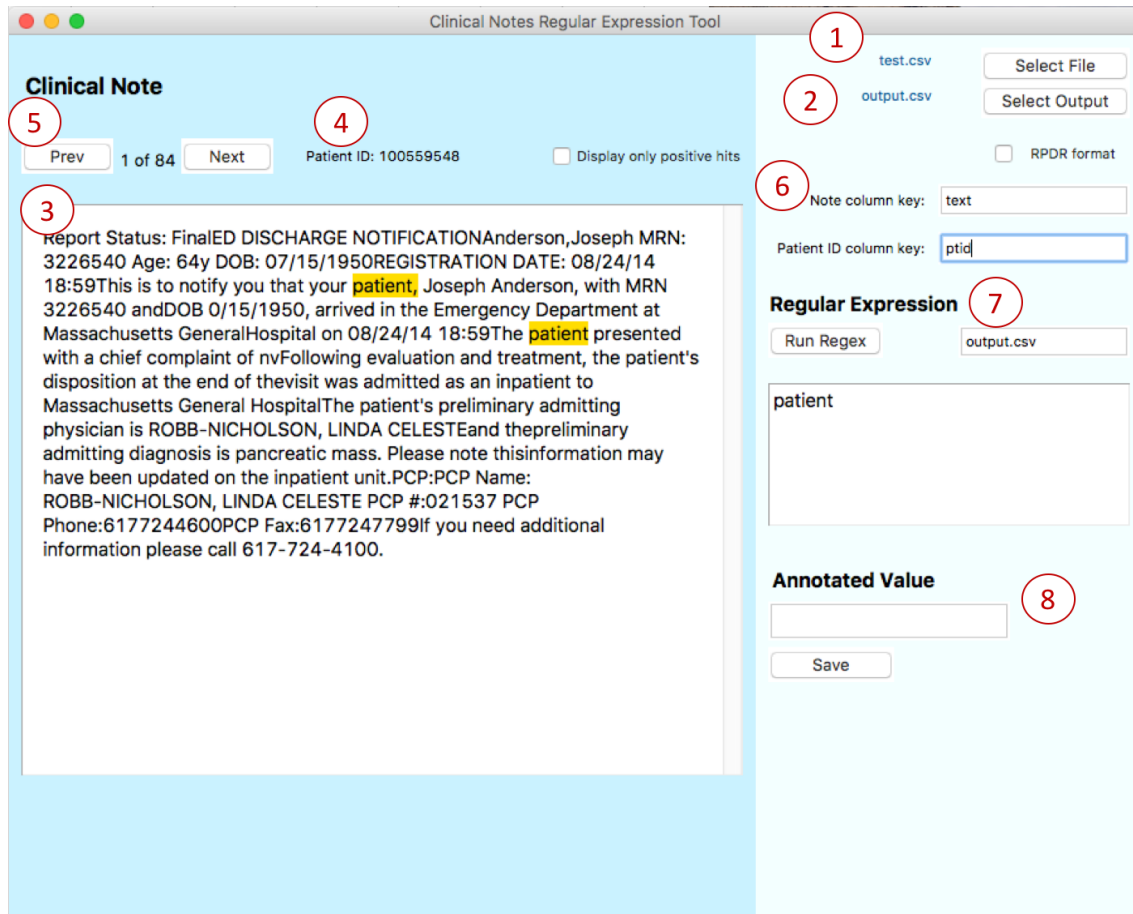


Figure 2-2: ClinicalRegex user interface elements. 1: File selection button. Can accept RPDR text files or any CSV file with note and ID columns. Displays file name here. 2: Output file selection button. The user can elect to view already created output files. When the user runs the regular expression on a selected file, the output file automatically appears here. 3: Clinical text box, where the clinical note is displayed with identified text highlighted in yellow. 4: Patient ID displayed here, or whatever column value is specified in the Patient ID column key. 5: Previous and Next buttons navigate to the previous or next note. 6: Text boxes where you can specify the column name where the clinical note and patient IDs reside. Must be correctly specified if the input file is not an RPDR file. Is hidden when RPDR format is selected. 7: Regular expression functionality occurs here. The user can specify any output file name they would like, the default is “output.csv.” They can enter comma-separated texts into the textbox. These are the keywords the regular expressions will extract. To run the regular expression on the input file, the user presses the “Run Regex” button. 8: Annotation text box. When the regular expression is complete, or if the user opens an output file, they can annotate each note with a value. When they press “Save” this value is written to file.

can be checked if the user only wishes to view notes that have a keyword identified.

There is also functionality for rudimentary user annotation of the clinical notes. A text box for “Annotated Value” accepts any character as an annotation, which is saved in the output file. If an annotated value is currently saved in the file, it is displayed in the text box. The “Save” button must be pressed in order for the value to be saved. Additionally, output files can be selected, viewed, and edited using the “Select Output” button.

2.4 Application to identifying seriously ill patients

ClinicalRegex was used in a study to identify seriously ill patients with understudied disease processes using a combination of administrative data and NLP.

Serious illness is defined as proposed by Kelley et al.: a health condition that carries a high risk of mortality and negatively impacts quality of life, or excessively strains caregivers[24]. Using a combination of administrative codes and NLP, we identified two cohorts of patients with serious illness that could not be isolated by standard methods alone. We chose to focus on one medical and one surgical diagnosis: pneumoperitoneum and leptomeningeal metastases secondary to stage IV breast cancer. Both disease processes are associated with high short-term mortality and often require a reevaluation of treatment priorities[2, 42, 46]. Since it is difficult to identify these patient populations, we have little understanding of who may benefit from an intensive, interventional approach and who may benefit from a comfort focused approach.

2.4.1 Data

Our primary data source was the Partners HealthCare Research Patient Data Registry. This registry gathers data from multiple EHRs at Partners HealthCare. Administrative data is available for encounters across all hospital and clinic settings within the Partners HealthCare system. Data are linked to EHR notes, including admission notes, consultation notes, progress notes, procedure notes, and discharge summaries.

Initial population screening was performed using ICD-9 diagnosis codes which are compiled for every patient encounter in Partners HealthCare. We used ICD-9 diagnosis codes for unspecified peritoneal disorder (568.89) and visceral perforation (596.83) to identify patients with possible pneumoperitoneum between 2010 and 2015. We used ICD9 codes for breast cancer (174.0-174.9, 175.0, and 175.9) and leptomenigeal disease (198.4) to identify patients with possible leptomenigeal metastases between 2010 and 2016.

2.4.2 Approach

Despite their high mortality and morbidity, the administrative codes associated leptomenigeal metastasis and pneumoperitoneum have a low specificity. This has limited prior studies to small single institution case series[2, 42, 46]. In order to perform larger studies using multi-institutional data without the aid of dedicated registries we used a three-step approach. This process is illustrated in Figure 2-3.

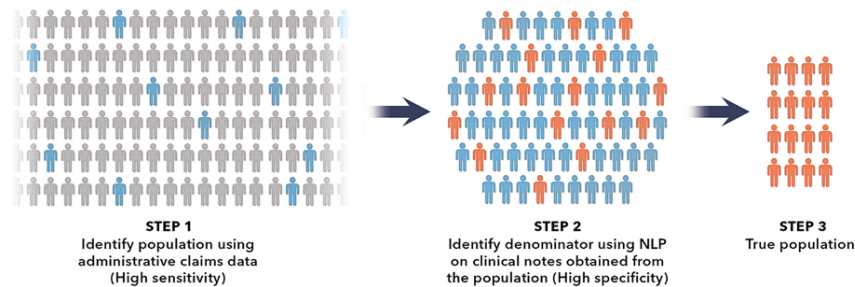


Figure 2-3: Three step process of patient identification

Step 1: We identified a large group of potential patients. For example, we used International Classification of Diseases, 9th Revision (ICD-9) diagnostic codes for unspecified peritoneal disorder and visceral perforation to screen for pneumoperitoneum. While the majority of patients who have these diagnoses code will not have pneumoperitoneum, it is likely that any patient that does have clinical relevant pneumoperitoneum will have one of these codes.

Step 2: From this pool of potential patients, ClinicalRegex allows for accurate and efficient isolation of the relevant population. We are able to scan through free-

text patient data (clinician notes, radiology reports, etc.) to identify keywords or phrases associated with a particular diagnosis. ClinicalRegex is used to identify patients with a specific disease. For pneumoperitoneum, we used the phrases free air, extra-luminal air, extra-luminal gas, and pneumoperitoneum to review radiology reports. For leptomenigeal disease, we reviewed radiology reports using the keyword leptomenigeal.

Step 3: The charts of patients identified through Step 2 are audited to ensure accuracy. The review process is performed in a semi-automated fashion in which the positive notes are listed with the positive phrase or word highlighted in order to facilitate identification. Using ClinicalRegex reduces the time required to review charts by 1,000 fold compared to manual chart review by physicians[33]. Semi-automated review ensures that the positive search terms have not been taken out of context. For example, regular expressions may erroneously identify a note in which the phrase no evidence of precedes the desired search term. Through machine learning the system can learn to distinguish between positive and negative indicators and eventually obviate the need for any review.

2.4.3 Results

Initial population screening using ICD-9 diagnosis codes yielded 6,438 patients in the with pneumoperitoneum and 557 with metastatic leptomenigeal disease. These patients hospitalization was associated with 299,449 and 32,519 radiology reports, respectively. Using regular expressions, we reduced the number of reports by approximately 95%. The reports were reviewed by trained physicians in a semi-automated fashion as described in Section 2.4.2. Through this process, we were able to rapidly, within hours, identify 869 patients with pneumoperitoneum and 187 patients with leptomenigeal metastasis. Use of administrative codes alone was associated with a positive predictive value (PPV) of 13% for pneumoperitoneum and 25% for leptomenigeal metastasis from breast cancer as compared against the human-validated notes. Those identified through the three step process utilizing administrative codes and regular expressions achieved a PPV of 100% in both cases, as compared to the

human-validated notes.

2.5 Application to assessment of palliative care processes

Several methodological barriers impede the development and implementation of quality measures in palliative surgery[40]. Prospectively gathering patient-reported outcomes after palliative surgery is costly, time-consuming, and hindered by high attrition due to serious illness and early mortality[31, 35]. As a result, the current literature on palliative surgery outcomes is largely limited to single-institution retrospective studies, which may not be generalizable and are influenced by regional variations in end-of-life care intensity[32]. Moreover, data extraction from chart review is laborious, subject to interpreter bias, and the use of diverse methodologies impedes extrapolation of findings across studies[3, 4, 29]. Multi-site and population-based data are needed, but the use of large databases is hindered by the limitations of administrative claims codes, which do not distinguish palliative vs. curative procedure indications, are not sensitive to post-operative changes in symptom burden, and do not describe palliative care processes, such as documentation of code status discussions before surgery.

The use of electronic health record (EHR) combined with natural language processing has the potential to overcome the challenges associated with assessing quality indicators. To address this major gap, we developed and tested NLP using existing data from EHR to: (1) retrospectively identify patients with malignant bowel obstruction who received palliative venting gastrostomy tube for refractory nausea and vomiting and, (2) develop and refine quality benchmarks for processes of care, such as documentation of pre-operative advance care planning.

2.5.1 Data source and study population

The primary data source was the Partners HealthCare Research Patient Data Registry. This data is linked to the EHR and include admission notes, consultation notes, progress notes, procedure notes, operative reports, and discharge summaries.

We used International Classification of Diseases, Ninth Revision, Clinical Modification (ICD9-CM) and Current Procedural Terminology (CPT) administrative codes to identify cancer patients (ICD9-CM 140-209) who received a gastrostomy tube (ICD9-CM 43.11, 43.19, 44.32 or CPT 49440) from January 1, 2012, to March 31, 2016. All clinical notes for these patients were obtained from the Research Patient Data Registry.

2.5.2 Methods

Administrative codes do not accurately identify whether the gastrostomy procedure was indicated for feeding or palliative venting; therefore, we used NLP to process textual data from the clinical notes. Specifically, we utilized regular expressions, which can identify patterns of characters exactly as they are specified, in the form of the ClinicalRegex software. ClinicalRegex identified patients who had the key word venting documented within one week of their procedure. Our final denominator was cancer patients who were treated with venting gastrostomy tube placement as identified by administrative code and NLP methods.

Additionally, ClinicalRegex was used to identify documentation of the following validated process measures: goals of care discussions, clarifying code status, palliative care consultation, and assessment for hospice. Our keyword library was then used to enumerate the instances of process documentation at three distinct time points in the procedural timeline: two months before gastrostomy tube placement, during admission for gastrostomy tube placement, and after the procedure admission. The percentage of patients with documentation at each of these three time points was then calculated to determine a process measure score.

Methodology for determining the indication for the gastrostomy procedure through

manual chart review has been previously described[31]. In short, a single researcher noted whether the gastrostomy was indicated for venting a malignant obstruction (palliative indication). A second researcher reviewed a 20% random sample of charts. Inter-rater agreement was excellent ($\kappa = 0.97$)[50].

To determine each end-of-life quality metric, the EHR of 20 randomly selected patients were manually reviewed by two human coders, providing a gold standard for NLP performance. We developed a code book to ensure gold-standard manual chart review, shown in Table 2.1. This rule book included definitions and keywords for each process measure. Each human coder identified clinical text that included one of the specified keywords, and annotated the clinical text to indicate which process measure it was associated with.

The human coders annotated all 1,710 clinical notes for 21 patients, 5 were coded by both. In the case of disagreement in the notes reviewed by both coders, a third human coder reviewed the specific note in question, and broke the tie between the original human coders.

2.5.3 Results

Using the keyword library (shown in Table 2.1) that we developed to assess the quality metrics, we used ClinicalRegex on the same notes from the 68 identified patients that were previously scored by human coders. The performance of regular expressions for quality measures is shown in Table 2.2. We identified care preferences, code status, palliative care and hospice discussions with high (85.7%, 90.8%, 92.9%, 89.6%, respectively) sensitivity and high (96.7%, 90.6%, 98.2%, 98.9%, respectively) specificity compared to human coders.

Table 2.1: Regular expression keyword library for assessment of palliative care processes

Process Measure	Key Words
<p>Clarifying code status: Conversations with patients or family members about preferences for cardiopulmonary resuscitation and intubation. Includes limitations on life-sustaining treatment and confirmation, by the patient or family, of full code status. Does not include presumed full code status or if obtained from other sources (i.e., review of records, according to team).</p>	<p>Limitations on code status: dnr, dnrdni, dni, do not resuscitate, do-not-resuscitate, do not intubate, do-not-intubate, chest compressions, no defibrillation, no endotracheal intubation, no mechanical intubation, shocks, cmo, comfort measures.</p> <p>Full Code Status: Full code confirmed, full code d/w, full code discussed, full code verified, would like to be full code, wishes to be full code, would like to remain full code, wishes to remain full code, wish to be full code, remaining full code, full code.</p> <p>MOLST</p>
<p>Goals of care discussions: Conversations with patients or family members about the patients goals, values, or priorities for treatment and outcomes. Includes statements that conversation occurred as well as listing specific goals.</p>	<p>Goals of care, goc, goals for care, goals of treatment, goals for treatment, treatment goals, family meeting, family discussion, family discussions, debility/goals of care, goc/coping, patient goals</p>
<p>Palliative care referral: Documentation that palliative care specialists were involved or that consultation was considered or offered, regardless of whether consultation occurred.</p>	<p>Pallcare, palliative care, pall care, pallcare, palliative medicine, supportive care</p>
<p>Hospice assessment: Documentation that hospice was discussed, prior enrollment in hospice, patient preferences regarding hospice, and assessments the patient did not meet hospice criteria.</p>	<p>Hospice</p>

Table 2.2: Performance of regular expression for identification of quality measures

	Process Measure	NLP Library	Sensitivity (95% CI)	Specificity (95% CI)
Denominator	Patient with cancer requiring placement of venting gastrostomy.	Venting	95.8% (88.4-97.0)	97.4% (94.5-99.1)
	Patients with billing codes for cancer AND gastrostomy procedure AND key word venting documented within one week of the procedure.			
Numerator	ASSIST: IF a patient is newly known to have advanced cancer after a surgery, diagnostic test, or physical exam, THEN a discussion including prognosis and advance care planning should be documented within 1 month or a reason given why such a discussion did not occur.	Goals of Care	85.7% (84.6-87.4)	96.7% (89.6-91.3)
		Code Status	90.8% (89.4-91.1)	90.5% (97.6-98.8)
	ASSIST: IF an outpatient with advanced cancer dies an expected death, THEN he or she should have been referred for palliative care within six months before death (hospital-based or community hospice) or there should be documentation why there was no referral.	Palliative Care	92.9% (91.8-94.2)	98.2% (97.6-98.9)
		Hospice	89.6% (88.2-91.0)	98.9% (98.2-99.3)

Chapter 3

PyCCI: A Clinical Note Annotation Tool

3.1 Introduction

Annotated clinical notes can be used for a variety of research topics and are especially essential in the field of natural language processing. Annotated notes can be used for rule-based methods as well as for deep learning and validation of both such methods. In this chapter, I present a clinical note annotation tool that I have developed for the purposes of natural language processing research applied to clinical subjects. PyCCI comes in two versions: PyCCI General and PyCCI Symptoms. PyCCI General allows for user-specified (in a configuration file) annotation domains. PyCCI Symptom is developed specifically for annotation of documentation of symptoms, and includes a few differences in features from PyCCI General. Both versions have been applied to machine learning research, which are described here and in Chapter 4.

3.2 Architecture

PyCCI is implemented using Python 3.5 and leverages Tkinter, a graphical user interface Python package. It is implemented using a modular architecture, with each class representing a different general function in the user interface. Figure 3-1 displays

architecture of this program. The program is compiled into an executable file using PyInstaller, which allows it to be easily utilized by someone unfamiliar with computer programming.

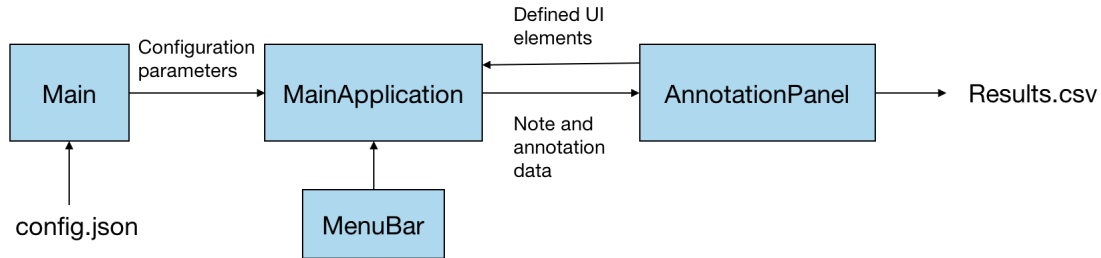


Figure 3-1: Software Architecture of PyCCI

The application is launched from main.py. In PyCCI General, main.py reads in config.json, a configuration file that can be edited by any user to reflect the parameters they wish to provide. Most significantly, they can decide what categories they would like to label text as. For example, for the advanced care planning project (described in Chapter 4), we used the following categories: “Patient and Family Care Preferences,” “Communication with Family,” “Full Code Status,” “Code Status Limitations,” “Palliative Care Team Involvement,” “Ambiguous.” In PyCCI Symptoms, the parameters are not configurable. Other differences in the two versions include handling of commas in text boxes, highlighting convention, and existence of related categories. These will be described further in Section 3.3. These required minimal technical alterations and do not affect the general architecture of the program.

There are two main modes of the program: annotation and reviewing. These workflows are determined based upon what button is used to open files: “Open CSV” or “Open Results” respectively. Data is handled using Pandas DataFrames. Files selected should be readable into Pandas DataFrames. Data remains in DataFrame format until it is saved, at which point the current DataFrame is written to an output csv file.

The MainApplication class is primarily responsible for defining the user interface with the specified configuration parameters and also maintains the data models. It additionally houses most of the functionality for defining and modifying what is

displayed in the interface, including event handlers and the display of highlighted text based on annotations. The highlighted text is clickable; clicking the highlighted text reveals options for editing the annotation. This functionality is also housed in `MainApplication` and is aided by classes `TagData`, `AnnotatorTagData`, and `ReviewTagData`.

The `AnnotationPanel` class defines the right-hand panel of the interface, where a textbox is displayed for each defined category. The separation of the `AnnotationPanel` from `MainApplication` allows for convenient modularity of the annotation functionality, which works well with the variable nature of annotation categories. `AnnotationPanel` creates the user interface objects for annotations: the text boxes and checkboxes that represent each category. It is also responsible for saving annotations, including processing and validating the text in the text boxes to ensure that it appears in the note, and then manipulating the format of the data so that it can be saved in the specified file format (as described in Section 3.3.3). This involves adding validated annotations to a new `DataFrame`, which is written to a results csv file.

The functionality for reviewing annotations from multiple annotators is unfortunately not designed in a modular fashion and is instead mixed in the `MainApplication` class. When files are selected for review, functions in `MainApplication` determine overlapping annotated elements and use that to determine how the text is then displayed (highlighted). The highlighted text is clickable, and the logic to determine what options are available are also defined in `MainApplication`.

In reviewing, the user has the option to alter existing annotations, delete existing annotations, or add new annotations. This functionality is fairly complex; when files for review are selected, a new csv file is created that assigns each unique interval of annotated text its own line. This file can be re-opened in review mode so that a reviewer can continue working from the same data. Unique intervals are defined as intervals that share the same labels. Let us consider the example “patient had full code status.” If the full text is labelled as `Label1` and only “patient” is labelled as `Label2`, then there would be two unique intervals: “patient” and “ had full code status.” This required development of an algorithm to quickly determine separate

intervals of annotated text. Modifications to the existing annotations are saved to this new file. The existing files are not modified, allowing us to preserve the integrity of that data.

When in review mode, the user is still able to add new annotations. This functionality is housed in AnnotationPanel. The user is also able to alter existing annotations. The functionality for that is in MainApplication, but the logic for saving any changes or new annotations is still in AnnotationPanel. Annotations added while in review mode are saved to the new file as mentioned in the previous paragraph.

3.3 Features and Functionality

PyCCI can be used entirely through the user interface. Figure 3-3 displays the different elements of the user interface in detail for PyCCI General, which are the same as for PyCCI Symptoms. Figure 3-4 displays the user interface in review mode for PyCCI Symptoms (which has the same functionality as PyCCI general reviewing). Figure 3-2 displays a flowchart of possible workflows.

As touched upon previously, PyCCI comes in two variations: General and Symptoms. General allows for user-defined configurations and categories. Symptoms is used specifically for annotation of documentation of symptoms.

PyCCI is an executable program that can be launched with a double click. It is paired with a config.json configuration file that allows the user to define certain parameters that are then reflected in the launched program. See Table 3.3 for a detailed description of available parameters. The program is then friendly to modification by non-technical users.

Table 3.1: PyCCI configuration parameters

Parameter	Description	PyCCI Symptoms Configuration
------------------	--------------------	-------------------------------------

<code>title</code>	String of title displayed at top of user interface	Heart Failure Symptoms
<code>text_config</code>	Dictionary of various configuration variables	See below
<code>keywords_fname</code>	Name of the keywords file that displays keywords in yellow highlight	keywords.txt
<code>text_key</code>	Column name of column in input file that contains clinical note texts	TEXT
<code>note_key</code>	Column name of column in input file that contains unique note ID	ROW_ID
<code>patient_key</code>	Column name of column in input file that contains patient ID	HADM_ID
<code>category_key</code>	Column name of column in input file that contains note category	CATEGORY
<code>results_fname_suffix</code>	String that is appended the end of the input filename. The result is the name of the output file.	Results.csv
<code>review_fname</code>	Filename for output of annotation mode.	reviewed.csv
<code>textbox_labels</code>	List of intended annotation categories. Displayed in list order	[“Negative Symptom”, “Negation”, “Positive Symptom”, “Positive Modifier”, “Neutral Symptom”, “Ambiguous”]

<code>checkbox_labels</code>	List of intended checkbox categories (annotation category that does not contain corresponding text box).	[“None”]
<code>commentbox_labels</code>	List of intended commentbox categories. Allows free text	[]
<code>textbox_label _to_keypress</code>	Dictionary of <code>textbox_label</code> to intended hotkey	{‘Negative Symptom’: ‘s’, ‘Negation’: ‘a’, ‘Positive Symptom’: ‘f’, ‘Positive Modifier’: ‘d’, ‘Neutral Symptom’: ‘g’, ‘Ambiguous’: ‘e’}
<code>textbox_label _to_code</code>	Dictionary of <code>textbox_label</code> to intended short string that represents that label in the output file	{‘Negative Symptom’: ‘NSY’, ‘Negation’:‘NEG’, ‘Positive Symptom’: ‘PSY’, ‘Positive Modifier’:‘PMO’, ‘Neutral Symptom’: ‘NEU’, ‘Ambiguous’: ‘AMB’}

3.3.1 Annotating

Annotation mode is begun when the user selects a file using the “Open CSV” button. The accepted file format is a CSV file with data added row-wise. Column keys can be specified in the configuration file as detailed in Table 3.3. Most significant are the `text_key` and `note_key`, which denote which columns contain the clinical text and a unique identifier for each note respectively.

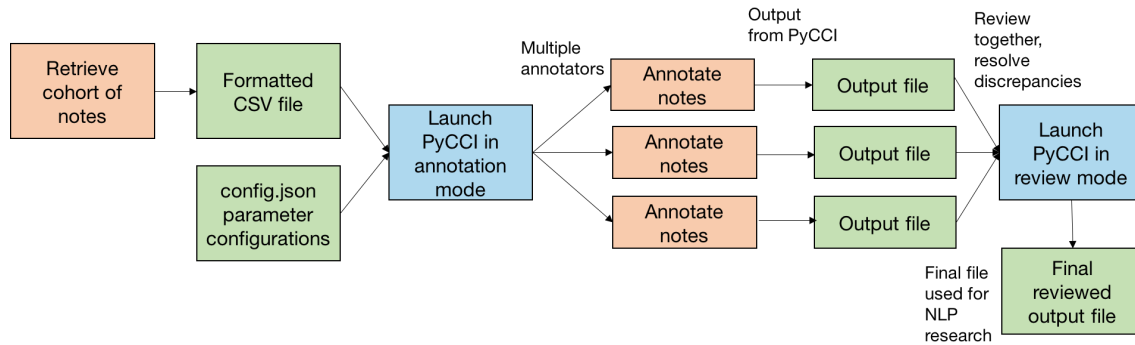


Figure 3-2: PyCCI typical workflow. Files are in green, actions in orange, PyCCI-use in blue.

If a file with a valid format is selected, the first note (row-wise) is displayed in the clinical text box. If a keywords file is specified, all words that appear on this list in the clinical text are highlighted in yellow. This can aid annotators in annotation, as they can see words that are likely to be relevant to the categories they are annotating.

Annotators and annotate pieces of text by highlighting that text, copying it, and then pasting it to the corresponding annotation category textbox on the right-hand panel of the interface. Hotkeys can also be used, as specified in the configuration file. If hotkeys are configured, a user can highlight text they wish to annotate and then press the hotkey for the corresponding annotation category. The highlighted text will then appear in the corresponding text box, eliminating the need for manual copy and pasting. In PyCCI Symptoms, items separated by commas are considered distinct entities and are saved in different rows of the output file.

The user can save these annotations by pressing the “Save Annotation” button or navigating to the previous or next note. If the user stays on the same note (by clicking “Save Annotation”), they will be able to see their annotations highlighted in the clinical text box, in blue. Pieces of text that are annotated with multiple labels are highlighted in a darker blue color. In PyCCI Symptoms, each category (Positive, Negative, or Neutral) are highlighted in a different color, with overlapping labels also being a darkened shade of the color. The user can delete their labels by clicking on the highlighted text; a menu will pop up and display options for deletion of a label. When annotations are saved, they are written to a new results file (the name of the

file can be specified in the configuration file). The format of this file is described in Section 3.3.3.

3.3.2 Reviewing

Review mode is begun with the user selects files using the “Open Results” button. The user can select any number of files to review. These files must be in the results format from annotation mode. All files are compared against each other to determine where annotators agree or disagree. A new file is created in the same folder where the files are selected from, with name as specified in the configuration file (and always named “reviewed.csv” in the Symptoms version). All notes are displayed, with texts with agreeing labels highlighted in green, texts with only one annotator having labelled it in red, and texts with disagreeing labels in red. See Figure 3-4 for an image of the review interface.

Modifications and additions can be made; the reviewer can modify existing annotations by clicking on the highlighted texts and choosing from a list of options (to add a label or delete labels). The reviewer can also add completely new annotations if they have noticed that all annotators have missed it. Everything is written to the new “reviewed.csv” file upon save.

3.3.3 File Formats

This section provides an overview of the files that are input or output from PyCCI and describes their formats.

- Annotation input file: Any csv file that can be read into a Pandas DataFrame can be accepted by PyCCI, as long as column keys are correctly specified as described in Table 3.3.
- Annotation output file: The output of annotation mode is a csv file (saved from a Pandas DataFrame) that has each piece of annotated text in a row, with position and label information, as well as any metadata about the note that was present in the original file.

- **Keywords file:** This file represents a list of texts that the user wants to be highlighted in yellow during the annotation process. Typically, this is text that is likely to belong in one of the annotation categories. The file should be provided in txt format with each string on a new line.
- **Review input files:** Review mode accepts any files in the annotation output file format. All files that the reviewer would like to compare can be selected at once.
- **Review output file:** The review output file is similar in format to the annotation output file. Upon launching the review software and selecting the files to review, the program creates a new “reviewed.csv” output file with each unique annotated interval of text on a new row. Modifications to the annotations result in modifications to this file.

3.4 Applications

PyCCI was used for the research in Chapters 4 of this thesis. As described in Chapter 4, PyCCI General was used to annotate advanced care planning documentation. Each note was annotated by two board-certified clinicians and finally reviewed by a third clinician, who was responsible for validating and tie-breaking the existing annotations. In order to ensure consistent annotation, a set of abstraction guidelines was developed for the annotators. Each annotator identified text from clinical notes that fit specified advanced care planning indicators. The portion of text that was deemed relevant to the indicator was labeled in its entirety, with no restrictions on length of a single annotation. Although the selected text was only used once for one indicator, the same text could be used for multiple indicators if deemed appropriate by the annotator. A similar process can be followed for annotation in completely different projects. The final domains can be implemented in the configuration file.

PyCCI Symptoms was also utilized in a project to annotate documentation of heart failure symptoms. Here too, each note was annotated by two board-certified

clinicians and finally reviewed by a third clinician, who was responsible for validating and tie-breaking the annotations. When this research was occurring, PyCCI General had been developed. Our team discovered certain traits specific to the documentation of symptoms that required a set of new features for convenience, eventually leading to the development of PyCCI Symptoms.

Deciding upon annotation domains for PyCCI Symptoms was an iterative process that involved many meetings and determining what would make sense for natural language processing methods to interpret. Ultimately, we decided upon six categories: negative symptoms, negative modifiers, positive symptoms, positive modifiers, and neutral symptoms and neutral modifiers. This provided needed granularity to our algorithms in development. Positive symptoms are symptoms that the patient has affirmed to have. Negative symptoms are those the patient denies having. Neutral symptoms are those such as “appetite,” which do not have a clear negative/positive dichotomy. Modifiers exist to provide more detail to the symptom. For example, a patient could indicate they have “more fatigue” with “more” being a positive modifier. While currently PyCCI Symptoms has only been applied to the identification of heart failure symptoms, the possibilities of research into symptoms documentation are immense. This software can be extensible into any sort of symptoms documentation research, not just to heart failure symptoms.

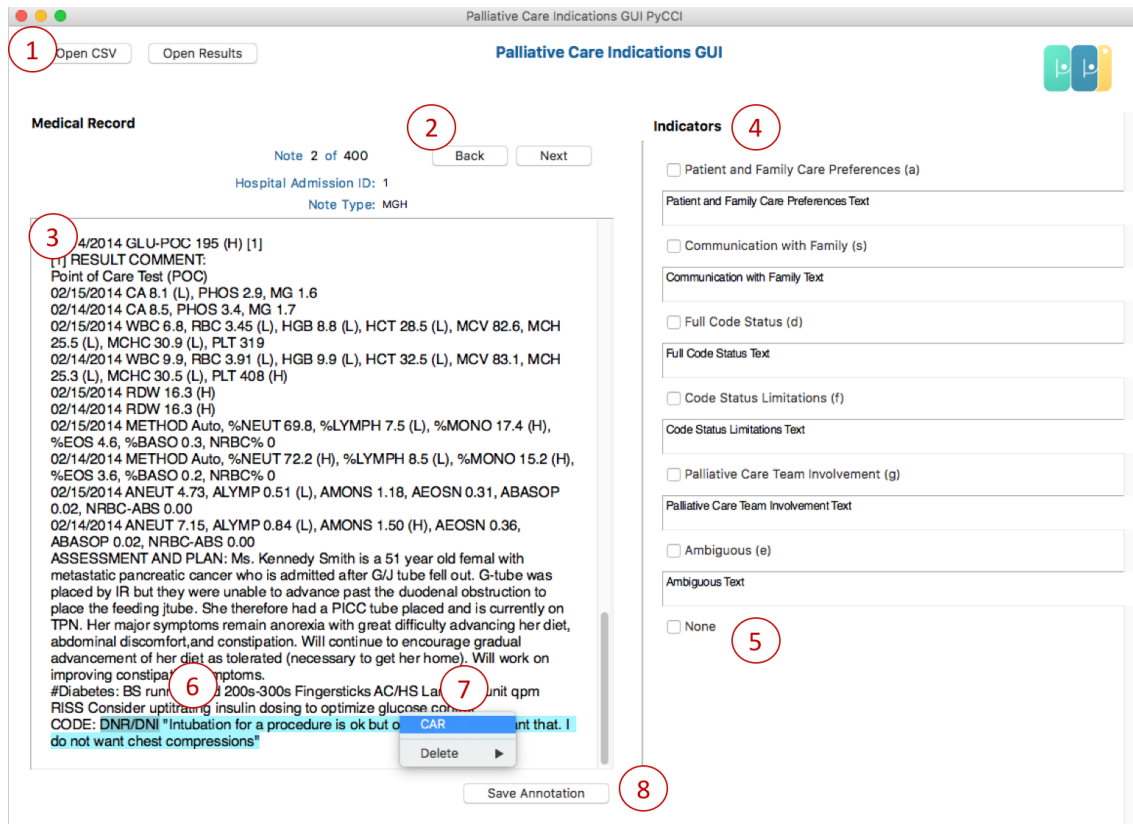


Figure 3-3: PyCCI General user interface elements. 1: File selection buttons. “Open CSV” opens a file for annotating, “Open Results” allows the user to select any number of files for review and comparison of completed annotations. 2: Back and Next buttons allows the user to navigate through notes. 3: Clinical notes text box, displays the current note. 4: Indicators are of the configured annotation categories. The user pastes text they wish to label into the corresponding box. The check box is automatically filled. 5: The “None” checkbox is used to indicate that a note has been reviewed and contains no labelled text. 6: Highlighted text. This is text that has been labelled and saved. The darker blue portion is text that has been labelled in multiple categories. 7: Popup menu on clicked highlighted text. Displays what category the text is labelled as and provides delete commands. 8: Save button. Pressed to save the annotations currently in the indicator text boxes.

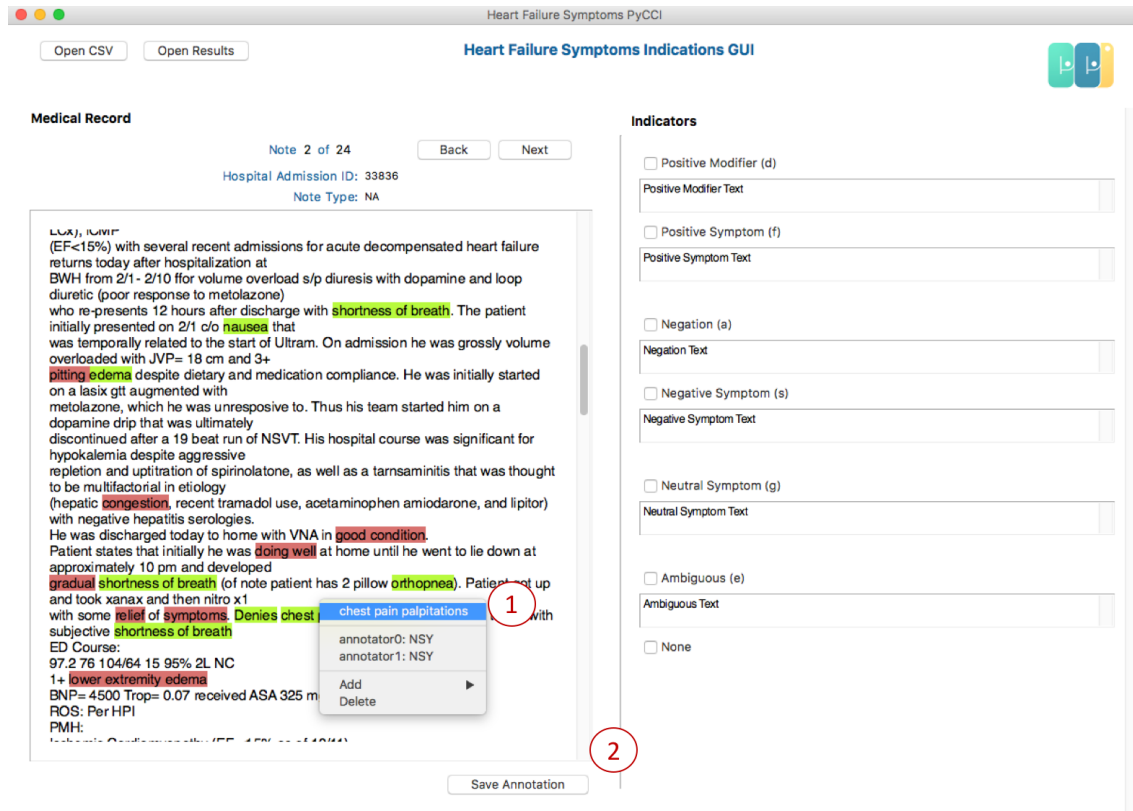


Figure 3-4: PyCCI Symptoms review mode user interface elements. 1: Popup menu on clicked highlighted text. Displays the corresponding text snippet, the labels assigned by annotators, and options to add or delete labels. You can see that labelled texts with agreeing labels from annotators are in green, while conflicting labels are in red. 2: Save button. This button saves the annotations to file.

Chapter 4

Natural Language Processing for Advanced Care Planning

4.1 Introduction and related work

As discussed in Chapter 1, advanced care planning, which includes clarifying and documenting goals of care and preferences for future care, is essential for achieving end-of-life care that is consistent with the preferences of dying patients and their families. Physicians document their communication about these preferences as unstructured free text in clinical notes; as a result, routine assessment of this quality indicator is time consuming and costly.

The emergence of omnipresent EHRs and powerful computers present novel opportunities to apply advanced computational methods such as natural language processing (NLP)[37] to assess end-of-life quality metrics including documentation of ACP. NLP enables machines to process or “understand” natural language in order to perform tasks like extracting communication quality embedded as non-discrete free-text within clinical notes[20].

Two main approaches to NLP information extraction exist. Rule-based extraction uses a pre-designed set of rules[37], which involves computing curated rules specified by experts, resulting in algorithms that detect specific words or phrases. This approach works well for smaller defined sets of data such as when searching for all the

brand names of a generic medication (e.g., if X is present, then $Y=1$). However, rule-based approaches fail when the desired information appears in a large variety of contexts within the free text[7].

Recent advances in machine learning coupled with increasingly powerful computers have created an opportunity to apply advanced computational methods, such as deep learning, to assess the content of free-text documentation within clinical notes. Such approaches possess the potential to broaden the scope of research on serious illness communication, and when implemented in real-time, to change clinical practice.

In contrast to rule-based methods, deep learning does not depend upon predefined set of rules. Instead, these algorithms learn patterns from a labeled set of free-text notes and apply them to future datasets[7]. A deep learning-based approach works well for tasks for which the set of extraction rules is very large, unknown, or both. In deep learning, algorithms can learn feature representations that aid in interpreting varied language.

In this study, we used deep learning[44] to train models to detect documentation of serious illness conversations, and we assess the performance of these deep learning models against manual chart review and rule based regular expression.

4.2 Data

4.2.1 Data Source

We derived our sample from the publicly available ICU database, Multi Parameter Intelligent Monitoring of Intensive Care (MIMIC) III, developed by the Massachusetts Institute of Technology (MIT) Lab for Computational Physiology and Beth Israel Deaconess Medical Center (BIDMC)[23]. It is a repository of de-identified administrative, clinical, and survival outcome data from more than 58,000 ICU admissions at BIDMC from 2001 through 2012. Between 2008 and 2012, the dataset also included clinical notes associated with each ICU admission. The Institutional Review Board of the BIDMC and MIT have approved the use of the MIMIC-III database by any

Table 4.1: Sample characteristics

General Note Statistics	Training Data Set	Validation Data Set
Number of notes	449	192
Number of tokens	612450	282788
Word count, mean (interquartile range), words	1362.2 (987.0 - 1664.0)	1472.9 (1045.0 - 1819.0)
Patient Demographics		
Number of unique patients	279	123
Age, mean (SD)	71.5 (14.4)	69.9 (15.9)
Female, Number (%)	136/279 (48.8)	52/123 (42.3)
Type of ICU at admission, Number (%) of notes		
Coronary Care Unit (CCU)	50/448 (11.2)	13/192 (6.8)
Cardiac Surgery Recovery Unit (CSRU)	14/448 (3.1)	6/192 (3.1)
Medical ICU (MICU)	313/448 (69.9)	127/192 (66.2)
Surgical ICU (SICU)	38/448 (8.5)	30/192 (15.6)
Trauma Surgical ICU (TSICU)	33 /448 (7.4)	16/192 (8.3)
Note-level Statistic by Domain, Number (%) of notes		
Patient care preferences	187/449 (41.6)	92/192 (47.9)
Goals of care conversations	129/449 (28.7)	74/192 (38.5)
Code status limitations	138/449 (30.7)	59/192 (30.7)
Communication with Family	171/449 (38.1)	86/192 (44.8)
Full code status	292/449 (65.0)	130/192 (67.7)

investigator who fulfills data-user requirements. The study was deemed exempt by the Partners Institutional Review Board.

4.2.2 Cohort

The study population included adult patients (age ≥ 18) who were admitted to the medical, surgical, coronary care, or cardiac surgery ICU. The training and validation set included physician notes from patients who died during the hospital admission to ensure that we would have sufficient examples of documentation of care preferences. We excluded patients who did not have physician notes within the first 48 hours because these patients either died shortly after admission or transferred out of the ICU. The study population is described in Table 4.1.

Table 4.2: Clinical domain specifications.

Domain	Definition
Patient care preferences	Fulfills criteria for goals of care conversations and/or code status limitations
Goals of care conversations	Explicitly shown preferences about the patients goals, values, or priorities for treatment and outcomes. Does NOT include presumed full code status or if obtained from other sources.
Code status limitations	Explicitly shown preference of patients care restricting the invasive care. Includes taken over preference from previous admission.
Communication with family	Explicit conversations held during ICU stay period with patients or family members about the patients goals, values, or priorities for treatment and outcomes.
Full code status	Explicitly or implicitly shown preference for full set of invasive care including intubation and resuscitation. Includes presumed full code status or if obtained from other sources.

4.2.3 Clinical domains

Our main outcome was to identify documentation of care preferences within 48 hours of an ICU admission in seriously ill patients. We aimed to detect the binary absence or presence of any clinical text that fit specified documentation of domains: patient care preferences (goals of care conversations or code status limitations), goals of care conversations, code status limitations, family communication (which included communication or attempt to communicate with family that did not result in documented care preferences), and full code status. The specifications of each domain are outlined (Table 4.2).

4.2.4 Annotation

We developed a set of abstraction guidelines to ensure reliable abstraction between annotators. Each annotator identified clinical text that fit specified communication domains and labeled the portions of text identified for a domain, with no restrictions on length of a single annotation.

A gold standard dataset, considered to contain true positives and true negatives,

was developed through manual annotation by a panel of four clinicians. Annotation was done using PyCCI, a clinical text annotation software developed by our team. Each note was annotated by at least two clinicians and annotations were then validated by a third clinician. Similar to previously published chart abstraction studies performed for this measure, the abstraction team had real-time access to a US board certified hospice and palliative medicine attending physician-expert reviewer, met weekly, and used a log to document common questions and answers to facilitate consistency[50, 49].

The clinician coders manually annotated an average of 239 notes each (SD, 196), for a total of 641 notes. Each note contained an average of 1397 tokens (IQR, 1004-1710). The inter-rater reliability among the four clinician annotators was kappa $>$ 0.65 at the note level for each domain. The performance of each clinician coder was varied—for example, they identified documentation of care preferences with a sensitivity ranging from 77-92% (in comparison to the final gold standard).

4.3 Methods

4.3.1 Pre-processing

Annotated notes were pre-processed for both rule-based regular expression and neural network methods. First, texts were cleaned to remove any extraneous spaces, lines, or characters. Each cleaned note was tokenized, which means it was split into identifiable elements—in this case, words and punctuation. We used the Python module spaCy in order to tokenize intelligently, based on the structure of the English language[?]. Labels were associated with individual tokens and datasets were split out by domain, as each method was run separately.

4.3.2 Regular expression

Our baseline model is a simple regular expression based on pre-curated rules for each domain. Appendix A.1 shows the rules used for each domain. To create the regex

library, we identified tokens that were sensitive and specific for each prediction task. We calculated sensitivity by evaluating the proportion of a token’s total number of occurrences that were labeled for each domain. We evaluated specificity by evaluating what proportion of a token’s total number of occurrences were in a note that was in an unlabeled note for each domain. A board-certified clinician used these data points—sensitivity, specificity, frequency that each token appeared on the labeled text and frequency in texts outside of the domain—and their clinical knowledge to generate a list of terms that would likely be generalizable.

Regular expressions identify patterns of characters exactly as they are specified in a set of rules. If text in the note matches a keyword in the regex library for the domain, it is labelled as positive for that concept. This method acts as a baseline to compare our algorithm against. We used a regular expression program, ClinicalRegex, also developed by our lab[34]. ClinicalRegex is easily accessible and intuitive to navigate, which makes it an efficient choice for groups that are not able to employ computer scientists[34].

4.3.3 Artificial neural network

Deep learning involves training a neural network to learn data representation and fulfill a specified task. We trained algorithms to identify clinical text documentation of serious illness communication. During the training process, the neural network learns to identify and categorize tokens (individual words and symbols) as belonging to each of the pre-specified domains and maximizes probability across predicted token labels[12]. Figure 4-1 shows the pipeline used in training and validating the neural network.

The specific neural network used, NeuroNER, was developed by Deroncourt et al. for the purpose of named-entity recognition[11]. NeuroNER has been evaluated for use in the de-identification of patient notes[12]. It allows for each token to be labelled only with a single label. However, tokens in our study were often associated with multiple labels. For example, a sentence could indicate that both communication with family occurred and that goals of care were discussed. In order to allow for

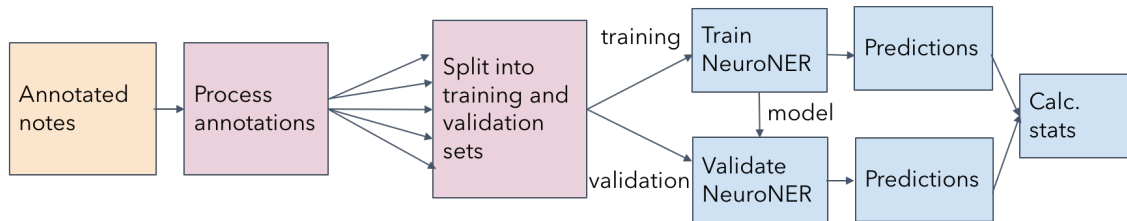


Figure 4-1: Pipeline used to train and validate neural networks to identify ACP documentation

multi-class labelling, a separate, independent model was trained per domain. For each domain, the data set was split up into randomized training and validation sets, with 70% (449 notes) of the set in training, and 30% (192 notes) in validation.

With the parameters derived from this training process, the model is run on the validation data set to examine its performance on a data set it was not specifically tuned to fit. Performance on the validation set also determines when training converges, indicating that the model is optimally trained. Training converges when there has been no improvement on the validation set performance in ten epochs. The neural network ultimately determines domain labels for each token. From the predicted token-level results, a note-level classification is determined by the presence or absence of labelled tokens by domain in each note. We used Tensorflow version 1.4.1 and trained our models on a NVIDIA Titan X Pascal GPU. Below are the hyperparameters selected for our use:

- character embedding dimension: 25
- character-based token embedding LSTM dimension: 25
- token embedding dimension: 100
- label prediction LSTM dimension: 100
- dropout probability: 0.5

For our experiments, we were able to compare our gold standard labels, derived from manual annotation by clinicians as described in Section 2.4, to the predicted output to evaluate the performance of the neural network and the regular expression method.

4.4 Results

4.4.1 Evaluation metrics

Algorithm performance was determined at two levels: token-level and note-level, referring to the binary absence or presence of a label at these levels. Token-level results are more specific and allow accurate identification of relevant text within clinical notes. Note-level results allow determination of whether documentation of communication occurred. At both of these levels, we calculated the following metrics: sensitivity, specificity, positive predictive value, accuracy, and F1-score. The F1-score is the harmonic average of positive predictive value and sensitivity. It allows us to determine the success of our algorithm both in identifying true positives as well as true negatives.

The 95% confidence intervals for all metrics were determined via bootstrapping[15]; each trained network model was validated for 1,000 trials in addition to the reported performance point. During each trial, a validation set of 192 notes was created by random sampling with replacement of the original validation set of 192 unique notes. This creates an approximate distribution of performance for the model. In basic bootstrap technique, the 2.5th and 97.5th percentiles of the distributions for each metric are taken as the 95% confidence interval[9].

4.4.2 Performance

Table 4.3 summarizes the performance of the regular expression method and Table 4.4 summarizes the performance of the neural networks in identifying documentation of serious illness communication at the note level, for each clinical domain, on the validation set. Figure 4-2 displays a comparison in the F1-scores for each domain. For identification of documentation of patient care preferences, the algorithm achieved an F1-score of 92.0 (95% CI, 89.1-95.1), with 93.5% (95% CI, 90.0%-98.0%) sensitivity, 90.5% (95% CI, 86.4%-95.1%) positive predictive value and 91.0% (95% CI, 86.4%-95.3%) specificity. For identification of family communication without documentation

of preferences, the algorithm achieved an F1-score of 0.91 (95% CI, 0.87-0.94), with 90.7% (95% CI, 86.0%-95.9%) sensitivity, 90.7% (95% CI, 86.5%-94.8%) positive predictive value and 92.5% (95% CI, 89.2%-97.8%) specificity. Token-level performance is displayed in Appendix A.2.

At the note-level, we have been able to achieve high accuracy for all domains and see that in the validation set, the neural network outperforms the regular expression method in every domain for F1-score, significantly so in identifying patient care preferences, goals of care conversations, and communication with family. These domains contain more complex and diverse language, which are successfully identified by the neural network. A static library is not able to capture the diversity in these domains, necessitating the use of machine learning.

Table 4.3: Performance (%) of the regular expression method on the validation data set.

Domain	F1-score	Accuracy	Sensitivity	Positive Predictive Value	Specificity
Patient care preferences	76.0	78.6	70.7	82.3	86.0
Goals of care conversations	37.2	57.8	26.1	64.9	87.0
Code status limitations	94.3	96.4	98.3	90.6	95.5
Communication with family	43.6	67.7	27.9	100.0	100.0
Full code status	90.9	88.5	84.6	98.2	96.8

4.4.3 Error analysis

A review of documentation that the neural networks identified as serious illness conversations that was not labeled serious illness conversations the gold standard (false positives) showed that the algorithm identified documentation that clinician coders missed. Though our gold standard was rigorously validated, there still remains room for human error. Comparing the identified text from the neural network and regular expression methods, we found that as expected, the neural network was able to

Table 4.4: Performance (%) of the neural networks on the validation data set. Values in parentheses are 95% confidence intervals.

Domain	F1-score	Accuracy	Sensitivity	Positive Predictive Value	Specificity
Patient care preferences	92.0 (89.1-95.1)	92.2 (89.6-95.1)	93.5 (90.0-98.0)	90.5 (86.4-95.1)	91.0 (86.4-95.3)
Goals of care conversations	85.7 (80.4-90.3)	89.1 (85.6-92.4)	85.1 (78.4-91.5)	86.3 (80.0-93.0)	91.5 (87.7-95.7)
Code status limitations	95.9 (93.0-98.7)	97.4 (95.8-99.2)	98.3 (96.9-100.0)	93.5 (89.2-97.7)	97.0 (95.0-98.9)
Communication with family	90.7 (87.4-93.9)	91.7 (89.1-94.4)	90.7 (86.0-95.9)	90.7 (86.5-94.8)	92.5 (89.1-95.9)
Full code status	98.5 (97.5-99.4)	97.9 (96.6-99.2)	100.0 (100.0-100.0)	97.0 (95.1-98.9)	93.5 (89.2-97.7)

identify complex and unique language that the regular expression method was not. Doctors employ diverse and unstandardized language in clinical notes; we require more flexible and extensible methods in order to efficiently process this information. Static libraries cannot capture the full complexity of language without sacrificing sensitivity or specificity—they must be curated such that library terms are not too broad and they are not able to utilize context. All note-level identification can be traced to the detection of specific words with examples of text for each method provided in Appendix A.3.

4.4.4 Effect of training set size

In order to determine how smaller training sets related to the performance of the trained algorithms, we trained multiple networks with varying number of notes. We plotted training dataset size against algorithm performance for 8 sample sizes (Figure 4-3). The performance seemed to plateau at around 200 notes (around 250,000 tokens), which suggests that annotation efforts can be efficiently leveraged to generalize the models to varied health systems.

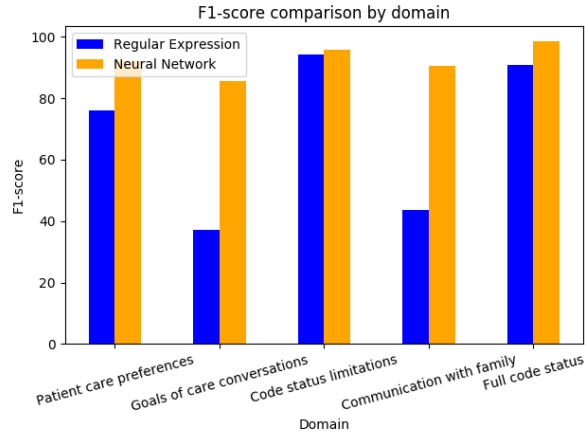


Figure 4-2: Comparison between the F1-score of the regular expression method and neural networks by domain.

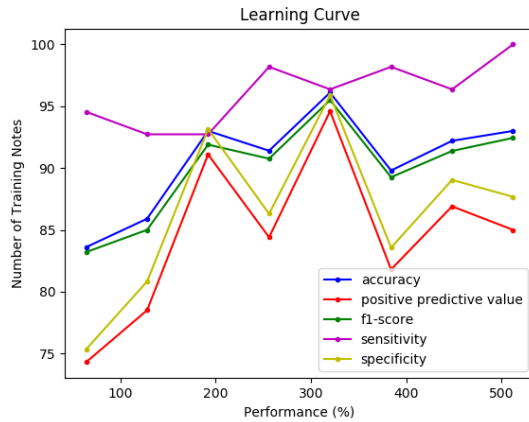


Figure 4-3: Neural network performance on validation set for detection of note-level documentation of patient care preferences by number of notes used for training.

4.5 Conclusions and future work

We describe a novel use of deep learning algorithms to rapidly and accurately identify documentation of serious illness conversations within clinical notes. When applied to identifying documentation of patient care preferences, our algorithm demonstrated high sensitivity (93.5%), positive predictive value (90.5%) and specificity (91.0%), with a F1-score of 92.0. In fact, we found that deep learning outperformed individual clinician coders both in terms of identifying the documentation and in terms of its many-thousands-time-faster speed.

Existing work has shown that machine learning can extract structured entities like medical problems, tests and treatments from clinical notes[10, 52], and unstructured image-based information in radiology, pathology and ophthalmology[18, 19, 48]. Our study extends this line of work and demonstrates that deep learning can also perform accurate automated text-based information classification.

Up until now, extracting goals of care documentation nested within free-text clinical notes has relied on labor-intensive and imperfect manual coding[50]. Using the capabilities of deep learning as demonstrated in this paper would allow for rapid audit and feedback regarding documentation at the system and individual practitioner level. This would result in significant opportunities for quality improvement that are currently not being met. Deep learning models could also improve patient care in real-time by broadening what is available at the point of care in the EHR. For example, clinicians could view displays of all documented goals of care conversations, or be prompted to complete documentation that was not yet available.

Important limitations must be noted. Deep learning algorithms only detect what is documented. It is not fully understood to what extent documentation reflects the actual content of a patient-clinician conversation surrounding serious illness care goals. However, documentation is the best proxy we have to understand and to track these conversations. This is also a single institution study, which may limit its generalizability. Future work will involve the investigation of how extensible models are to clinical notes from different health system. Variations in EHR software and the structure of clinical notes in different institutions makes it essential to further train and validate our methods using data from multiple healthcare systems. This should be imminently possible, as our learning curve suggested that the neural network needed to train on as few as 200 clinician coded notes to perform well. Future research should also focus on optimizing deep neural networks to further improve performance, and on determining the feasibility of operationalizing this algorithm across institutions.

This is the first known report of employing deep learning, to our knowledge, to identify serious illness conversations. The potential of this technology to improve the visibility of documented goals of care conversations within the EHR and for quality

improvement has far reaching implications. We hope such methods will become an important tool for evaluating and improving the quality of serious illness care from a population health perspective.

Appendix A

Tables

A.1 Regular expression library

Domain	Keywords
Patient care preferences	goc, goals of care, goals for care, goals of treatment, goals for treatment, treatment goals, family meeting, family discussion, family discussions, patient goals, dnr, dni, dnrdni, dnr/dni, DNI/R, do not resuscitate, do-not-resuscitate, do not intubate, do-not-intubate, chest compressions, no defibrillation, no endotracheal intubation, no mechanical intubation, shocks, cmo, comfort measures
Goals of care conversations	goc, goals of care, goals for care, goals of treatment, goals for treatment, treatment goals, family meeting, family discussion, family discussions, patient goals
Code status limitations	dnr, dni, dnrdni, dnrdni, DNIR, do not resuscitate, do-not-resuscitate, do not intubate, do-not-intubate, chest compressions, no defibrillation, no endotracheal intubation, no mechanical intubation, shocks, cmo, comfort measures
Communication with family	Explicit conversations held during ICU stay period with patients or family members about the patients goals, values, or priorities for treatment and outcomes.
Full code status	full code

A.2 Token-level performance

Performance (%) of the neural network on the validation data set at the token-level.

Domain	F1-score	Accuracy	Sensitivity	Positive Predictive Value	Specificity
Patient care preferences	76.0	99.6	75.8	75.2	99.8
Goals of care conversations	70.4	99.6	70.0	69.9	99.8
Code status limitations	76.3	99.8	72.7	80.5	99.9
Communication with family	68.2	99.7	62.0	76.4	99.9
Full code status	90.9	99.8	88.3	93.6	99.8

A.3 Examples of identified text

Below are examples of correctly identified serious illness documentation by the neural network and regular expression methods in the validation dataset. Correctly identified tokens are bolded. Typographical errors are from the original text. Each cell includes an example of identified tokens in the same text and an example of documentation identified by the neural network that was missed by the regular expression method, if relevant.

Domain	Neural Network	Regular Expression
--------	----------------	--------------------

Goals of care conversations	<p>Hypercarbic resp failure: family meeting was held with son/HCP and in keeping with patients goals of care, there was no plan for intubation.Family was brought in and we explained the graveness of her ABG and her worsened mental status which had failed to improve with BiPAP. Family was comfortable with removing Bipap and providing comfort care including morphine prn.</p> <p>family open to cmo but pt wants full code but also doesn't want treatment or to be disturbed.</p>	<p>Hypercarbic resp failure: family meeting was held with son/HCP and in keeping with patients goals of care, there was no plan for intubation.Family was brought in and we explained the graveness of her ABG and her worsened mental status which had failed to improve with BiPAP. Family was comfortable with removing Bipap and providing comfort care including morphine prn.</p> <p>family open to cmo but pt wants full code but also doesn't want treatment or to be disturbed.</p>
Code status limitations	<p>CODE: DNR/DNI, confirmed with healthcare manager who will be discussing with official HCP</p>	<p>CODE: DNR/DNI, confirmed with healthcare manager who will be discussing with official HCP</p>

<p>Communication with family</p>	<p>Dr. [**First Name (STitle) **] from neurosurgery held family meeting and explained grave prognosis to the family.</p> <p>lengthy discussion with the son who is health care proxy he wishes to pursue comfort measures due to severe and unrevascularizable cad daughter is not in agreement at this time but is not the proxy due to underlying psychiatric illness</p>	<p>Dr. [**First Name (STitle) **] from neurosurgery held family meeting and explained grave prognosis to the family.</p> <p>lengthy discussion with the son who is health care proxy he wishes to pursue comfort measures due to severe and unrevascularizable cad daughter is not in agreement at this time but is not the proxy due to underlying psychiatric illness</p>
<p>Full code status</p>	<p>Code: FULL; Discussed with daughter and HCP who says that patient is in a Hospice program with a "bridge" to DNR/DNI/CMO, but despite multiple conversations, the patient insists on being full code</p> <p>CODE: Presumed full</p>	<p>Code: FULL; Discussed with daughter and HCP who says that patient is in a Hospice program with a "bridge" to DNR/DNI/CMO, but despite multiple conversations, the patient insists on being full code</p> <p>CODE: Presumed full</p>

Bibliography

- [1] Melissa D. Aldridge and Diane E. Meier. It is possible: quality measurement during serious illness. *JAMA Internal Medicine*, 173(22):2080, Sep 2013.
- [2] Brian Badgwell, Barry W. Feig, Merrick I. Ross, Paul F. Mansfield, Sijin Wen, and George J. Chang. Pneumoperitoneum in the cancer patient. *Annals of Surgical Oncology*, 14(11):31413147, Jul 2007.
- [3] Brian Badgwell, Robert Krouse, Janice Cormier, Caesar Guevara, V. Suzanne Klimberg, and Betty Ferrell. Frequent and early death limits quality of life assessment in patients with advanced malignancies evaluated for palliative surgical intervention. *Annals of Surgical Oncology*, 19(12):36513658, Jun 2012.
- [4] Brian D. Badgwell, Thomas A. Aloia, John Garrett, Gabe Chedister, Tom Miner, and Robert Krouse. Indicators of symptom improvement and survival in inpatients with advanced cancer undergoing palliative surgical consultation. *Journal of Surgical Oncology*, 107(4):367371, Jun 2012.
- [5] D. W. Bates, S. Saria, L. Ohno-Machado, A. Shah, and G. Escobar. Big data in health care: Using analytics to identify and manage high-risk and high-cost patients. *Health Affairs*, 33(7):11231131, Jan 2014.
- [6] Sara Bennett, Irene Akua Agyepong, Kabir Sheikh, Kara Hanson, Freddie Ssen-gooba, and Lucy Gilson. Building the field of health policy and systems research: An agenda for action. *PLoS Medicine*, 8(8), 2011.
- [7] David S Carrell, Robert E Schoen, Daniel A Leffler, Michele Morris, Sherri Rose, Andrew Baer, Seth D Crockett, Rebecca A Gourevitch, Katie M Dean, Ateev Mehrotra, and et al. Challenges in adapting existing clinical natural language processing systems to multiple, diverse health care settings. *Journal of the American Medical Informatics Association*, 24(5):986991, 2017.
- [8] Deborah Cook and Graeme Rocker. Dying with dignity in the intensive care unit. *New England Journal of Medicine*, 370(26):25062514, 2014.
- [9] A. C. Davison and D. V. Hinkley. Preface. *Bootstrap methods and their application*, pages ix–x, 1997.

- [10] Leonard W Davolio, Thien M Nguyen, Sergey Goryachev, and Louis D Fiore. Automated concept-level information extraction to reduce the need for custom software and rules development. *Journal of the American Medical Informatics Association*, 18(5):607613, 2011.
- [11] Franck Deroncourt, Ji Young Lee, and Peter Szolovits. Neuroner: an easy-to-use program for named-entity recognition based on neural networks. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2017.
- [12] Franck Deroncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, 2016.
- [13] KM Detering, AD Hancock, MC Reade, and W. Silvester. The impact of advance care planning on end of life care in elderly patients: randomised controlled trial. *BMJ*, 340:c1345, 2010.
- [14] Sydney M. Dy, Karl A. Lorenz, Sean M. Oneill, Steven M. Asch, Anne M. Walling, Diana Tisnado, Anna Liza Antonio, and Jennifer L. Malin. Cancer quality-assist supportive oncology quality indicator set. *Cancer*, 116(13):32673275, 2010.
- [15] B. Efron. Better bootstrap confidence intervals. Jan 1984.
- [16] National Quality Forum. Nqf 1626: Patients admitted to icu who have care preferences documented. *National Quality Forum*.
- [17] Lucy Gilson, Kara Hanson, Kabir Sheikh, Irene Akua Agyepong, Freddie Ssen-gooba, and Sara Bennett. Building the field of health policy and systems research: Social science matters. *PLoS Medicine*, 8(8), 2011.
- [18] Jeffrey Alan Golden. Deep learning algorithms for detection of lymph node metastases from breast cancer. *JAMA*, 318(22):2184, Dec 2017.
- [19] V. Gulshan, L. Peng, M Coram, and et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Journal of the American Medical Association*, 316(22):2402–2410, 2016.
- [20] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.
- [21] Thanh Huynh, Prince Raj, Eric Kleerup, and Neil Wenger. The opportunity cost of futile treatment in the intensive care unit. *Critical Care Medicine*, 42(9):1922–82, September 2014.

- [22] Thanh N. Huynh, Eric C. Kleerup, Joshua F. Wiley, Terrance D. Savitsky, Diana Guse, Bryan J. Garber, and Neil S. Wenger. The frequency and cost of treatment perceived to be futile in critical care. *JAMA Internal Medicine*, 173(20):1887, Nov 2013.
- [23] Alistair E.w. Johnson, Tom J. Pollard, Lu Shen, Li-Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, Roger G. Mark, and et al. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.
- [24] Amy S. Kelley and Evan Bollens-Lund. Identifying the population with serious illness: The denominator challenge. *Journal of Palliative Medicine*, 21(S2), 2018.
- [25] Nita Khandelwal, Erin K. Kross, Ruth A. Engelberg, Norma B. Coe, Ann C. Long, and J. Randall Curtis. Estimating the effect of palliative care interventions and advance care planning on icu utilization. *Critical Care Medicine*, 43(5):1102–1111, 2015.
- [26] Robert S. Krouse. Surgical palliation at a cancer center. *Archives of Surgery*, 136(7):773, Jan 2001.
- [27] H. M. Krumholz. Big data and new knowledge in medicine: The thinking, training, and tools needed for a learning health system. *Health Affairs*, 33(7):11631170, Jan 2014.
- [28] AC Kwok, Y-Y Hu, and et al. Invasive procedures in the elderly after stage iv cancer diagnosis. *The Journal of surgical research*, 193(2):754–763, 2015.
- [29] Alvin C Kwok, Marcus E Semel, Stuart R Lipsitz, Angela M Bader, Amber E Barnato, Atul A Gawande, and Ashish K Jha. The intensity and variation of surgical care at the end of life: a retrospective cohort study. *The Lancet*, 378(9800):14081413, 2011.
- [30] Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and H. V. Jagadish. Regular expression learning for information extraction. *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP 08*, 2008.
- [31] Elizabeth J. Lilley, Alexandra B. Columbus, and Zara Cooper. Inferring palliative intent from administrative data: Validation of a claims-based case definition for venting gastrostomy tube. *Journal of Pain and Symptom Management*, 53(6), 2017.
- [32] Elizabeth J. Lilley, Kashif T. Khan, Fabian M. Johnston, Ana Berlin, Angela M. Bader, Anne C. Mosenthal, and Zara Cooper. Palliative care interventions for surgical patients. *JAMA Surgery*, 151(2):172, Jan 2016.

- [33] Elizabeth J. Lilley, Charlotta Lindvall, Keith D. Lillemoe, James A. Tulsy, Daniel C. Wiener, and Zara Cooper. Measuring processes of care in palliative surgery. *Annals of Surgery*, page 1, 2017.
- [34] Charlotta Lindvall, Elizabeth J. Lilley, Sophia N. Zupanc, Isabel Chien, Alexander W. Forsyth, Anne Walling, Zara Cooper, and James A. Tulsy. Natural language processing to assess palliative care processes in cancer patients receiving palliative surgery. In preparation., 2018.
- [35] Jennifer W. Mack, Angel Cronin, Nancy L. Keating, Nathan Taback, Haiden A. Huskamp, Jennifer L. Malin, Craig C. Earle, and Jane C. Weeks. Associations between end-of-life discussion characteristics and care received near death: A prospective cohort study. *Journal of Clinical Oncology*, 30(35):43874395, Oct 2012.
- [36] Laurence E. Mccahill, Robert Krouse, David Chu, Gloria Juarez, Gwen C. Uman, Betty Ferrell, and Lawrence D. Wagman. Indications and use of palliative surgery-results of society of surgical oncology survey. *Annals of Surgical Oncology*, 9(1):104112, 2002.
- [37] Genevieve B. Melton and George Hripcsak. Automated detection of adverse events using natural language processing of discharge summaries. *Journal of the American Medical Informatics Association*, 12(4):448457, 2005.
- [38] Thomas J. Miner. The palliative triangle. *Archives of Surgery*, 146(5):517, Jan 2011.
- [39] TJ Miner, MF Brennan, and DP Jaques. A prospective, symptom related, outcomes analysis of 1022 palliative procedures for advanced cancer. *Ann. Surg.*, 240(4):719–726, 2004.
- [40] Travis B. Murdoch and Allan S. Detsky. The inevitable application of big data to health care. *Jama*, 309(13):1351, Mar 2013.
- [41] Lauren Hersch Nicholas, Kenneth M. Langa, Theodore J. Iwashyna, and David R. Weir. Regional variation in the association between advance directives and end-of-life medicare expenditures. *JAMA*, 306(13):1447, May 2011.
- [42] Anna Niwiska, Katarzyna Pogoda, Wojciech Michalski, Micha Kunkiel, and Agnieszka Jagieo-Gruszfeld. Determinants of prolonged survival for breast cancer patient groups with leptomeningeal metastasis (1m). *Journal of Neuro-Oncology*, 138(1):191198, Dec 2018.
- [43] JC. Rising and T Valuck. Building additional serious illness measures into medicare programs. *The Pew Charitable Trusts*, 2017.
- [44] Jrgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85117, 2015.

- [45] Kabir Sheikh, Lucy Gilson, Irene Akua Agyepong, Kara Hanson, Freddie Ssen-gooba, and Sara Bennett. Building the field of health policy and systems research: Framing the questions. *PLoS Medicine*, 8(8), 2011.
- [46] Myrick C. Shinall, Jesse M. Ehrenfeld, and Oliver L. Gunter. Nonoperative management of perforated hollow viscera in a palliative care unit. *Annals of Surgery*, page 1, 2018.
- [47] JM Teno, A Gruneir, Z Schwartz, A. Nanda, and Wetle T. Association between advance directives and quality of endoflife care: A national study. *Journal of the American Geriatrics Society*, 55(2):189–194, 2007.
- [48] DSW Ting, C.Y. Cheung, G. Lim, G.S.W. Tan, N.D. Quang, A Gan, H Hamzah, R Garcia-Franco, IY San Yeo, and SY. Lee. Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. *Journal of the American Medical Association*.
- [49] AM Walling, S.M. Asch, K.A. Lorenz, C.P. Roth, T. Barry, K.L. Kahn, and N.S. Wenger. The quality of care provided to hospitalized patients at the end of life. *Archives of Internal Medicine*, 170(12):1057–1063, 2010.
- [50] Anne M. Walling, Diana Tisnado, Steven M. Asch, Jennifer M. Malin, Philip Pantoja, Sydney M. Dy, Susan L. Ettner, Ann P. Zisser, Hannah Schreibeis-Baum, Martin Lee, and et al. The quality of supportive cancer care in the veterans affairs health system and targets for improvement. *JAMA Internal Medicine*, 173(22):2071–2079, Sep 2013.
- [51] AA Wright, B. Zhang, and et al. Associations between end-of-life discussions, patient mental health, medical care near death, and caregiver bereavement adjustment. *JAMA*, 300(14):1665–1673, April-June 2008.
- [52] Hua Xu, Min Jiang, Matt Oetjens, Erica A Bowton, Andrea H Ramirez, Janina M Jeff, Melissa A Basford, Jill M Pulley, James D Cowan, Xiaoming Wang, and et al. Facilitating pharmacogenetic studies using electronic health records and natural-language processing: a case study of warfarin. *Journal of the American Medical Informatics Association*, 18(4):387391, 2011.