5-2024

Low-Resource ICD Coding of Discharge Summaries

Ashton Williamson
taw2@clemson.edu

Low-resource ICD Coding of Discharge Summaries

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science

---

by
Timothy Ashton Williamson
May 2024

Accepted by:
Dr. Amy Apon, Committee Chair
Dr. Nina Hubig
Dr. Brian Dean

Abstract

Medical coding is the process by which standardized medical codes are assigned to patient health records. This is a complex and challenging task that typically requires an expert human coder to review health records and assign codes from a classification system based on a standard set of rules. Considering the downstream use of these codes in statistical analysis, billing, and patient care, improving the accuracy and efficiency of the medical coding process through automation could have a far-reaching impact on the healthcare domain. Since health records typically consist of a large proportion of free-text documents, this problem has traditionally been approached as a natural language processing (NLP) task. While machine learning-based methods have seen recent popularity on this task, they tend to struggle with codes that are assigned less frequently, for which little or no training data exists.

In this thesis, we utilize the open-source programming language for natural language processing, NLP++, and its associated integrated development environment to design and build an automated system to assign International Classification of Diseases (ICD) codes to discharge summaries that functions in the absence of labeled training data. We evaluate our system using the MIMIC-III dataset and compare our results to supervised approaches. Results show that for datasets where labels are sparse, our approach matches state-of-the-art machine learning approaches. It is somewhat less effective for densely labeled datasets, but provides additional sup-

port for explainability and adaptability. Overall, our approach presents an effective pathway for code assignment in clinical documents by providing both competitive performance and enhanced explainability.

Dedication

Acknowledgments

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Medical coding is the process by which health clinics and institutions assign standardized codes to patient health records for downstream use in applications such as statistical analysis, indexing patient health records, coding medical billing claims [32], and assessing quality of patient care [40]. In the United States, The Health Insurance Portability and Accountability Act (HIPAA), enacted in 1996, mandates the use of standard coding systems for healthcare transactions and delegates oversight of the execution of the standardization process to the department of Health and Human Services [1, 37]. At the time of writing, the coding systems that have been selected for use include both the Clinical Modification (ICD-10-CM) and Procedure Coding System (ICD-10-PCS) of the 10th revision of the International Classification of Diseases, the Current Procedural Terminology (CPT), the Healthcare Common Procedure Coding System (HCPCS), the Code on Dental Procedures and Nomenclature (CDT), and the National Drug Codes (NDC) [38]. Each coding system is specific to a particular subset of medical terminology and healthcare application and,

due to the exacting nature of the medical lexicon, each can be quite large, with the ICD-10 comprising more than 70,000 codes and the HCPCS comprising over 7,000 codes. While these systems of classification represent a critical infrastructure with extensive significance in the healthcare domain, the success of their implementation largely rests on the efficient and precise assignment of these codes to a patient's health record. In practice, however, the task of assigning ICD codes to patient health records is a complex, multi-stage process with many possible points of failure [40]. While clinicians ultimately generate the *paper trail* that makes up the health record for a patient hospital visit and have some involvement in the assignment of medical codes, much of the burden of this process is placed on trained human experts, known as *medical coders*.

Despite the widespread adoption of electronic health records systems [35] and the proliferation of software to aid in the parsing of medical data [2], medical coding remains a difficult task, even for expert coders. One challenge is that health records include a large amount of heterogeneous data, including clinical notes, lab reports, tabular data, and imaging results, requiring medical coders to synthesize information across a broad range of modalities. To add to the complexity, constituent elements of the health record are typically recorded at different phases of a patient's trajectory and by different clinicians, introducing variations in terminology and notation [40]. Additionally, coders are required to have a working knowledge of various classification systems, each containing thousands of codes, as well as the complex rules governing their assignment [12]. In practice, this leads to considerable variation in the assignment of medical codes [12] and undermines the objective of standardization for which these classification systems were designed. The compounded effect of this variation in the assignment of medical codes can adversely affect downstream applications which utilize these codes. Particularly noteworthy is the financial impact

resulting from their use in medical billing. From a financial perspective, increased automation in medical encoding could reduce the need for human coders and increase encoding accuracy, both of which would reduce the cost burden of coding for healthcare providers and further limit their risk of fraud litigation [6], which can result in significant financial and institutional penalties. In fact, the Department of Health and Human services found in 2010 that approximately half of all claims for evaluation and management services were incorrectly coded, resulting in $6.7 billion in improper payments by Medicare [27]. Medical coding thus presents itself as a critical task which would greatly benefit from increased automation and could have a sweeping impact on healthcare, a sector for which national spending is projected to grow at an annual average rate of 5.4% per year to 2028 [16].

Since much of the information required for assigning codes is contained within unstructured text documents, the problem of medical coding has traditionally been approached through the framework of natural-language processing (NLP). Though much research has been conducted in this area, it remains a challenging problem and inherent limitations of state-of-the-art approaches tend to restrict their practical utility [12]. One such limitation of these approaches is a lack of explainability, or the ability to understand a system's decision-making process. This is especially detrimental to deep learning based approaches, which often function as *black-boxes*, providing little or no rationale for results. Current approaches to medical coding automation lack sufficient accuracy to be effectively employed in the clinical setting. Additionally, both classical machine-learning an deep-learning based approaches tend to be limited the need for quality data for initial training and validation, as well as ongoing testing and updating. Due to restrictions on the distribution of patient medical data, collecting and curating useful data sets for these tasks is a major challenge [22, 46, 23] and annotation costs to develop gold-standard datasets can be prohibitive [46].

To address the limitations of current approaches to the task of medical code assignment, we investigate the feasibility of applying a natural language processing programming language and integrated development environment called NLP++. NLP++ utilizes a *multi-pass, multi-strategy* architecture in which each pass over the input text performs a specific step in processing or parsing the text [10]. Furthermore, the multi-pass strategy constructs a single parse tree which is refined by each successive pass. In this architecture, passes are chosen or defined by the programmer to function according to the needs of the project. Sequences of passes are grouped together as an *analyzer* tailored to a particular application. Thus NLP++ can be adapted to a variety of natural language understanding tasks and downstream applications. NLP++ is selected due its support for explainability, adaptability, and its practicality in settings with little or no annotated data.

1.2 Research Challenges

In our analysis of the work on International Classification of Diseases coding we identify a few key areas of research which are both of critical importance and continue to pose challenging problems for existing systems: low-resource performance, explainability and adaptability. While each of these has been investigated to some degree in prior work (see Chapter 3), existing approaches fail to address these issues in a single, cohesive manner. Here we define each of these issues.

Another critical area of research which is under-studied is that of low-resource medical coding. A significant amount of research into ICD coding is conducted using the MIMIC [23] datasets, which provide clinical notes and corresponding ICD codes from the electronic health record system of the Beth Israel Deaconess Medical Center. It is worth noting that considerable effort was expended to both de-identify this data

and to aggregate and publish it [23]. While this constitutes a tremendous step forward for research, there is no guarantee that results on MIMIC are generalizable to other domains and assembling a dataset of this size and quality is not always feasible in a practical setting. Additionally, due to the long-tail distribution of code labels in MIMIC, some ICD codes are vastly underrepresented in the dataset and others don't appear at all, adversely affecting the overall performance of systems which rely on large amounts of training data [54]. For these reasons, few-shot (access to only a small set of training samples) and zero-shot (no access to in-domain training samples) approaches to ICD coding are of critical importance, but also an area in which existing state-of-the-art approaches tend to struggle [12, 54].

An explainable system provides a sensible, human-interpretable motivation for a particular decision, or a "trace of their inference steps" [18]. With respect to the coding of clinical notes, a popular approach to enhancing explainability is by providing a context span from the clinical note to support a particular code assignment. For example, for the code with title "Tobacco Use Disorder" a supporting text span from the note may be "... patient is a current smoker". These context spans allow for enhanced interoperability with human coders by providing a human-verifiable justification for a code assignment. Since automated systems for medical coding have not reached the performance level necessary to be deployed autonomously, these systems are typically used to support human coders. As such, the ability to provide human-interpretable predictions is of critical importance. Explainability poses a significant challenge for deep learning-based approaches.

Finally, we seek to address the issue of adaptability, a term which we use to describe a system which is able to be modified to address either the requirements of the user or changes in the input or output space. One drawback to existing state-of-the-art deep learning approaches to ICD coding is their inflexibility [12]. This

can be problematic for end-users of the system, who may wish to alter aspects of the deployed system to line up with individual or institutional requirements. This can also problematic with respect to the changing nature of coding systems like the ICD-9, in which codes are periodically removed, added, merged, or shifted. An ideal classification system would be able to easily adapt to these changes without the need for expensive re-training or significant architectural modifications.

The remainder of this thesis is organized as follows. In Chapter 2 , we provide background information on the research problem, including an overview of the MIMIC-III dataset, the International Classification of Diseases, NLP++ and the UMLS. In Chapter 3, we give an overview of previous work in the area of medical coding. Chapter 4 provides a detailed outline of the experiments conducted and Chapter 5 provides the results of these experiments, along with analysis of the results. Finally, we identify several promising directions for future work in Chapter 6.

Chapter 2

Background

2.1 MIMIC

A critical component in the development and deployment of automated systems for medical coding is annotated data. While sufficient data is being collected as a result of the increasing use of electronic health record systems, barriers remain to accessing and aggregating this data [35]. For one, the private nature of patient health records entails strict regulation and oversight, restricting access to and dissemination of patient medical records [22]. In addition, there is no incentive for EHR system vendors or medical institutions to facilitate the sharing of data with other organizations [25]. These restrictions inhibit the creation and curation of large, multi-institution health data sets, which are crucial for reducing biases and supporting more general AI technologies. Some work has been done to improve the availability of clinical data for research [23, 36], but there still exists a massive disparity between the amount of data available to researchers and that collected by health institutions [22]. One of the projects seeking to overcome this disparity is the *Medical Information Mart for Intensive Care*, or MIMIC [23, 21]. MIMIC is an openly accessible database of

de-identified electronic health record data for patients admitted to the intensive care unit of the Beth Israel Deaconess Medical Center, the teaching hospital of Harvard Medical School [23]. There are four releases of the MIMIC dataset (I-IV); our work focuses on the third release, since full access to the fourth release was not available until late in the project.

The third release of MIMIC, or MIMIC-III, was published in 2016 and contains data collected from 2001 to 2012 for 49,785 distinct hospital admissions corresponding to 38,597 unique patients [23]. Each hospital admission record is comprehensive and includes data falling into one of the following categories: billing, descriptive, dictionary, interventions, laboratory, medications, notes, physiologic, and reports [23]. In particular, MIMIC-III includes free text notes and reports–for example, radiology reports and hospital discharge summaries–as well as medical codes–ICD-9, CPT, and Diagnosis-Related Group (DRG) codes–for a hospital admission, making it an invaluable resource for the development and evaluation of automated code extraction systems.

Despite the demonstrated utility of MIMIC-III, the generalizability of results on the MIMIC-III coding dataset is limited by the quality of the assigned codes. ICD codes available in the MIMIC database, which are used as labels for the task of medical coding, are drawn from the electronic health record system [22]. This suggests that they have been assigned through the standard medical coding process, which is problematic given the typical incidence of error in medical code assignment [40, 12] and the frequent treatment of MIMIC-III ICD labels as a *gold standard* label set in the literature [46]. Searle et al. [46] conduct a review of ICD code assignment in MIMIC-III by comparing the labels in the dataset to entities found by a medical named-entity recognition and linking model, MedCAT. They find that the most frequently assigned codes in MIMIC-III are under-coded by up to 35% [46].

	Full [33]	**Top 50** [33]	**Rare 50** [54]
Number of Documents	52,723	11,368	391
Number of Patients	41,126	10,356	386
Number of Unique Codes	8,922	50	50
Mean Codes per Document	15.9	5.7	1.0
Train/Dev/Test %	90.5/3.1/6.4	71.0/13.8/15.2	60.6/4.9/34.5

Table 2.1: Subsets of the MIMIC-III coding dataset.

This is a significant result, since supervised approaches which rely on these labels can inherent these flaws. While a revision of code assignments in MIMIC-III via secondary validation would be a useful solution to this problem, it is an expensive solution [46] and one that has not yet been undertaken at scale. We aim to circumvent this issue by designing a system that functions in the absence of training data, referred to in the literature as a *zero-shot* setting.

2.2 International Classification of Diseases

The International Classification of Diseases constitutes a standardized nomenclature and classification system for diseases and medical procedures, which was originally intended to facilitate the statistical analysis of health data [32]. As the ICD has evolved and expanded, so have its practical applications in the healthcare domain, which have come to include the indexing of health record data in hospitals, the coding of medical billing claims [32], and the assessment of quality of patient care [40]. Each successive revision to the ICD, typically spanning 10-20 years, has sought to address these new use cases while simultaneously adapting to advances in medicine and healthcare [32]. As a result, the International Classification of Diseases has continued to grow in number of total codes as well as hierarchical depth, with the

ICD-9-CM–a clinical modification of the ICD-9 adapted for use in the US–containing over 14,000 codes and a maximum height of 6.

Each entity within the ICD-9-CM is encoded by a unique identification string consisting of three to five digits and an optional single letter prefix corresponding to a supplementary category (E for External Causes of Injury and Poisining, V for Factors Influencing Health Status and Contact with Health Services, and M for Morphology of Neoplasms) [40]. Each additional digit in a code specifies a greater specificity, or depth, within the hierarchy (see figure 2.1 for an example of a path within the ICD-9 hierarchy). In codes with more than three digits, a decimal point is added to distinguish between the three-digit parent class and the following digits.

One drawback to the approach to encoding hierarchical information in ICD-9 is the lack of correlation between the numerical representation of an entity in the ICD and its relative location in the hierarchy. For example, given that 140 is the 3-digit code for the category *Malignant Neosplasm of the Lip* and that 157 is the 3-digit code for the category *Malignant Neoplasm of Pancreas*, one might assume that the first digit, 1, corresponds to the more general class of *neoplasms*, while the last two digits correspond to severity or locality. Though the latter tends to hold, the former does not. Thus there is no way to determine from the code alone that code 135, *Sarcoidosis*, does not belong to the class *Neoplasms* (codes 140 to 239.99) but in fact belongs to *Infectious and Parasitic Diseases*, which spans codes 001 to 139.99.

More challenging than the structure of ICD codes are the challenges inherent to medical classification systems. One major issue that arises is that of synonymy. O'Malley et al. point out the example of *stroke*, a term with a large number of synonyms; in this case, one doctor could code cerebrovascular accident (Code 436) and another could code intracerebral hemmorhage (Code 431) and both would be correct. This adversely affects precision and could be a confounding factor for auto-

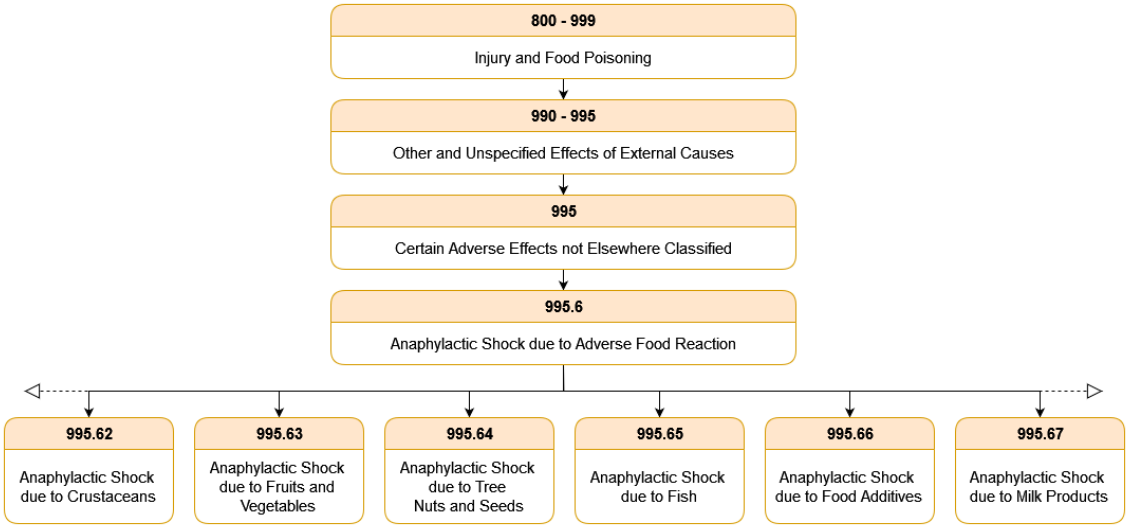

Figure 2.1: Path in the ICD-9 hierarchy for terminal child codes of 995.6: *Anaphylactic Shock due to Adverse Food Reaction*.

mated coding systems [40]. Another issue is the size and complexity of the ICD and the associated rules for code assignment. Some of the problems with synonymy and ambiguity in the ICD-9 could be resolved by more precise rules and more specificity in the hierarchy, but this would also have the effect of increasing the complexity of an already complex classification system. These limitations of the ICD serve to make ICD coding a particularly challenging task, especially when paired with complex and often ambiguous health record data. Some of these issues were considered in the development of the ICD-10. In particular, the coding system was redesigned to incorporate single letter prefixes corresponding to chapters as opposed to the arbitrary numeric ranges of the ICD-9. To address the issue of synonymy, titles were altered to incorporate a higher level of specificity in an attempt to better encapsulate the information needed to make a code assignment [52]. Additionally, the ICD-10 was expanded to allow for more granularity in the coding system, resulting in approximately 55,000 more codes compared to ICD-9. While these changes addressed critical issues

in the ICD-9, it is unclear to what extent ICD coding has improved in practice since the roll out of ICD-10. In 2013, Topaz et al [52] carry out an analysis of the literature on clinical data quality in the ICD-9 versus the ICD-9 and find no significant indication of "improvements or deficits". Our work focuses on the ICD-9 due to access to labeled ICD-9 data, however, we suspect that the methods outlined in this paper would carry over to ICD-10, either by restructuring the system to accommodate the new code set or by translating the ICD-9 output codes to ICD-10 codes using, for example, the general equivalence mappings provided by the Department of Health and Human Services [37].

2.3 NLP++

The technology selected for this project is a programming language dedicated to natural language processing, called NLP++ [30, 10]. NLP++ utilizes a *multi-pass, multi-strategy* architecture in which each pass over the input text performs a specific step in processing or parsing the text. Passes are broken down into specific regions: rule regions, code regions and declarative regions. Rule regions perform operations on the parse tree using predefined operators (for an example, see listing 2.3), code regions include code which is executed at runtime, and declarative regions include user-defined functions which can be called from both rule and code regions. The multi-pass strategy constructs a single, best-first parse tree which is refined by each successive pass. In the multi-pass architecture, passes are chosen or defined by the programmer to function according to the needs of the project. Sequences of passes are grouped together as an *analyzer* tailored to a particular application. NLP++ can thus be adapted to a variety of language tasks including information extraction, named entity recognition, sentiment analysis, word-sense disambiguation, and part of

speech tagging.

```
@NODES _ROOT

@RULES

_BLANKLINE <-
    _xWILD [min=0 max=0 matches=(\  \t \r)] ### (1)
    \n                                      ### (2)
    @@

_LINE <-
    _xWILD [min=0 max=0 fails=(\r \n)]      ### (1)
    _xWILD [one match=(\n _xEND)]           ### (2)
    @@
```

Listing 2.1: Code example of a built-in rule pass in an NLP++ analyzer to reduce lines in the parse tree. Blank lines are placed under a new node named *_BLANKLINE* and all other lines are placed under the *_LINE* node.

NLP++ also incorporates a hierarchical knowledge base management system (KBMS) called Conceptual Grammar (CG) for integrating domain, task, dictionary, and linguistic knowledge, as well as allowing the programmer to dynamically store and use information extracted from multiple input texts. The conceptual grammar includes both knowledge bases (files with extension *.kb*) and dictionaries (extension *.dict*). Knowledge bases allow the user to store and retrieve information in a hierarchical manner. Dictionaries, on the other hand, consist of entries and corresponding key/value pairs. After tokenization, a lookup is performed on the parse tree and nodes that match dictionary entries are tagged with their respective key/value pairs. This facility thus constitutes a key aspect of building effective analyzers for parsing

text.

```
ICD_9_CM
  800-999: title="Injury and Food Poisoning"
    990-995: title="Other and Unspecified Effects of External Causes"
      995: title="Certain Adverse Effects not Elsewhere Classified"
        995.6: title="Anaphylactic Shock due to Adverse Food Reaction"
          995.60: title="Anaphylactic shock due to unspecified food"
          995.61: title="Anaphylactic shock due to peanuts"
          995.62: title="Anaphylactic shock due to crustaceans"
          995.63: title="Anaphylactic shock due to fruits and vegetables"
          995.64: title="Anaphylactic shock due to tree nuts and seeds"
          995.65: title="Anaphylactic shock due to fish"
          995.66: title="Anaphylactic shock due to food additives"
          995.67: title="Anaphylactic shock due to milk products"
          995.68: title="Anaphylactic shock due to eggs"
          995.69: title="Anaphylactic shock due to other specified food"
```

Figure 2.2: Example of an NLP++ knowledge base for ICD code example from figure 2.1 with code ranges .

An integrated development environment (IDE) called the NLP Engine bundles both NLP++ and CG into a *sandbox* for rapid prototyping and development. VisualText supported deployments by NASDAQ and IBM Global Services UK, among others. The NLP Engine IDE supports multi-platform deployments on Linux, Windows, and Mac operating systems. The IDE serves to streamline and accelerate the prototyping, development, and maintenance of NLP systems in particular. NLP++ is also integrated into High Performance Computing Cluster (HPCC) Systems from LexisNexis for handling large datasets.

2.4 Biomedical Knowledge Bases

While the International Classification of Diseases represents an international standard in morbidity and mortality reporting, it is not the only attempt to system-

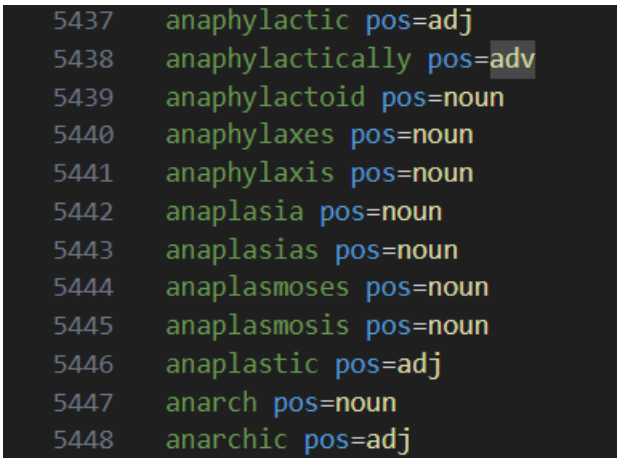

Figure 2.3: Section of the built-in part of speech dictionary for the English language.

atize biomedical knowledge. There exist a variety of medical coding systems, thesauri, and ontologies, each with their own terminology, structure, and scope. A key goal in biomedical research has thus been to develop systems to integrate these knowledge bases by developing a standard terminology and knowledge representation. One effort in this direction is the Unified Medical Language System (UMLS), a repository of biomedical vocabularies and associated tools which is developed and maintained by the US National Library of Medicine [4]. The UMLS consists of the Metathesaurus, a biomedical thesaurus which links concepts from different constituent vocabularies; the Semantic Network, which defines semantic types and provides relationships between UMLS concepts; and the SPECIALIST Lexicon, an English dictionary that includes biomedical terms [39].

.

The Metathesaurus is a collection of *source vocabularies* which includes biomedical thesauri, classification systems, coding systems, and controlled term lists [39] such as SNOMED-CT[50]. Terms in these vocabularies are linked to standard *identifiers* using semantic or lexical information. These identifiers, in order of increasing specificity, are Concept Unique Identifiers (CUIs) which refer to a specific meaning of

Concept (CUI)	Terms (LUIs)	Strings (SUIs)	Atoms (AUIs) * RRF Only
C0004238 Atrial Fibrillation (preferred) Atrial Fibrillations Auricular Fibrillation Auricular Fibrillations	**L0004238** Atrial Fibrillation (preferred) Atrial Fibrillations	**S0016668** Atrial Fibrillation (preferred)	**A0027665** Atrial Fibrillation (from MSH) **A0027667** Atrial Fibrillation (from PSY)
		S0016669 (plural variant) Atrial Fibrillations	**A0027668** Atrial Fibrillations (from MSH)
	L0004327 (synonym) Auricular Fibrillation Auricular Fibrillations	**S0016899** Auricular Fibrillation (preferred)	**A0027930** Auricular Fibrillation (from PSY)
		S0016900 (plural variant) Auricular Fibrillations	**A0027932** Auricular Fibrillations (from MSH)

Figure 2.4: An example of the different UMLS identifiers for terms mapping to the C0004238 concept: Atrial Fibrillation [39]. The table has columns for concept unique identifier, lexical unique identifier, string unique identifier and atom unique identifier.

a term; Lexical Unique Identifiers (LUIs) which link all lexical variants of a term, for example spelling variations, abbreviations or plural forms; String Unique Identifiers (SUIs) which are assigned to a specific string literal for each vocabulary; and Atom Unique Identifiers (AUIs) which are assigned to each individual entry in a vocabulary[39]. An example for the concept Atrial Fibrillation is provided in figure 2.4. The Semantic Network defines semantic types for these concepts as well as relationships between them. The Semantic Network defines 127 different semantic types and 54 different relationships, thus comprising a network in which types are the nodes and relationships are the vertices [39]. We leverage the this concept structure along with the relationships defined in the Semantic Network to map key terms found in

the text to concepts in the UMLS.

The Specialist Lexicon is a dictionary containing both common English terms as well as domain-specific biomedical terms which was developed with the express purpose of aiding in natural language processing [39]. The lexicon includes key linguistic information for each term, including spelling variations, conjugations or conjugation patterns, plural forms, and more [5]. Additionally, the NLM releases a set of utilities for working with the specialist lexicon. Of these utilities, we use the Lexical Variant Generator to generate variants and semantic neighbors of terms within the ICD-9 code titles.

Chapter 3

Related Work

Though research on automated medical coding dates as least as far back as the 1970's [42, 49], access to data and hardware limitations prevented the development of large-scale solutions. The first work on International Classification of Diseases coding specifically was published in the 1990s and treated the task as one of information retrieval, employing k-nearest-neighbors, relevance feedback, and Bayesian classifiers to select and rank relevant codes [26]. Other early approaches leveraged biomedical natural language processing systems, such as MedLEE [14], to extract clinically-significant entities which could then be linked to codes from the target coding system [3, 15]. While these approaches saw some success on test datasets, they were limited by their ability to generalize to new datasets and their ability to scale to larger label spaces.

Medori and Fairon examined the automated assignment of ICD-9-CM codes to French language clinical notes [29]. The express purpose of this study was to develop a tool to help medical coders at the *Cliniques Universitaires Saint-Luc*, thus the target corpus consisted of 166,670 notes and corresponding codes from the hospital's health record system. Despite its size, code-coverage was limited with only 4,039

unique codes present in the dataset. Their system was bipartite, including an extraction step using both dictionary-based and heuristic methods to identify relevant coding information and a classification step using Naïve Bayes classifiers to assign codes. Classifiers were built for codes that appeared more than five times in the corpus, resulting in only 1,497 classifiers. Results were limited in scope; the reported metrics were recall at 10, 15 and 20, and the highest of these, 81.1, was achieved after reducing classification to categories as opposed to individual codes. The approach taken by Medori and Fairon of separating the task into an extraction and classification step inspires our approach to isolating relevant context which is then used for code classification.

Perotte et al. utilize the hierarchical structure of the ICD-9 to aide in the assignment of ICD-9 codes to notes in MIMIC II [41]. Support-vector machine classifiers are trained for each ICD-9 code, including both terminal and parent codes, using a training set of MIMIC-II discharge summaries (20,533 total notes) [41]. At test time, notes in the test split (2,282 total notes) are evaluated from the root of the hierarchy downward such that each parent code serves as a gate for its respective child nodes. Thus a negative prediction for a node results in negative predictions for each of its descendants in the hierarchy. This approach demonstrates considerable performance improvement compared to a *flat* setup, in which predictions are made for each code without any consideration for hierarchy [41]. The hierarchical approach has the additional benefit of computational improvement at inference time, since negative predictions for parent codes effectively prune their subtrees in the hierarchy. Though the approach taken by Perotte et al. fundamentally differs from our approach (see Chapter 4), our approach to evaluating our system in a hierarchical setup is motivated by their success in exploiting the ICD hierarchy.

More recently, Chen et al. [9], investigated the feasibility of various NLP

systems for extracting codes from a subset of documents considered in the medical billing process including: Chief Complaint, History of Present Illness, and Past, Family, and/or Social History. The authors tested two systems on this task: a pipeline built around a rule-based system called CLAMP and a large language model trained on medical data, ClinicalBERT. The results demonstrated that each of the systems performed better at different tasks; for example, the rule-based system was better at identifying multi-word elements when considering the small training corpus, whereas the deep-learning model performed better at overall precision, but lacked the ability to effectively generalize due to the low amount of training data. Overall performance for both systems was low, suggesting that automating code extraction for medical billing still poses a challenging problem. Furthermore, the dataset used for training and testing consisted of only 61 samples (80% were used for training and 20% for testing) [9], which is likely too small to extrapolate the results to medical coding in general.

Mullenbach et al. [34] introduce an attentional convolutional network to assign ICD-9 codes to discharge summaries in the MIMIC-III dataset. Mullenbach et al. introduce train, development and test splits for the full set of MIMIC-III discharge summaries as well as splits for the top-50 subset which includes only the 50 most frequently assigned codes. Their system applies per-label attention to the output of a convolutional neural network and then applies a softmax function to the resulting vectors. For classification, these vectors are passed to an additional layer and sigmoid [34]. The advantage of this approach is that the intermediate output of the softmax can be interpreted as relative importance of locations in the document for a particular code assignment, thus enhancing the explainability of the approach while simultaneously achieving state-of-the-art performance. Both the full and top-50 train/development/test splits defined by Mullenbach et al. have become the standard

for comparison in the literature [54], so we choose to evaluate our system on the test split of each of these.

Yang et al. [54] address the long-tail challenge of ICD coding by both defining a rare code subset of the MIMIC-III dataset and introducing a training algorithm to improve performance on rare codes. The rare code subset, named *MIMIC-III-rare50*, includes the 50 least common codes in MIMIC-III and corresponding discharge summaries. The motivation for this subset comes from the observation that 4,115 of the 8,692 unique codes in the MIMIC-III dataset occur fewer than 6 times [54]. Given the reliance of pretrained language models on labeled data, few- and zero-shot settings like those introduced in the rare-50 subset pose a challenging problem. We use the rare-50 split to evaluate the performance of our system in a low-resource setting.

Chapter 4

Methods

4.1 MIMIC-III Dataset Pre-processing

The MIMIC-III dataset consists of a series of tables containing electronic health record data indexed by unique identifiers such as SUBJECT_ID, which refers to a unique patient, and HADM_ID, which corresponds to a unique hospital admission [23]. In particular, the NOTEEVENTS table contains charted notes along with corresponding metadata; for a comprehensive list of data types included in the NOTEEVENTS table, see Table 4.1. We extract all notes from the NOTEEVENTS table in the MIMIC-III dataset (version 1.4) with CATEGORY field matching *Discharge Summary*, including both reports and addenda. Notes where the ISERROR flag is set are dropped and all addendum-type discharge summaries are concatenated to their corresponding original reports, following previous work [33]. ICD-9 codes for these discharge summaries are then generated from the PROCEDURES_ICD and DIAGNOSES_ICD tables using the subject and hospital admission IDs. To facilitate processing, these codes are then mapped to their ICD-9 source representation by adding a decimal after the first two characters for procedure codes and after the

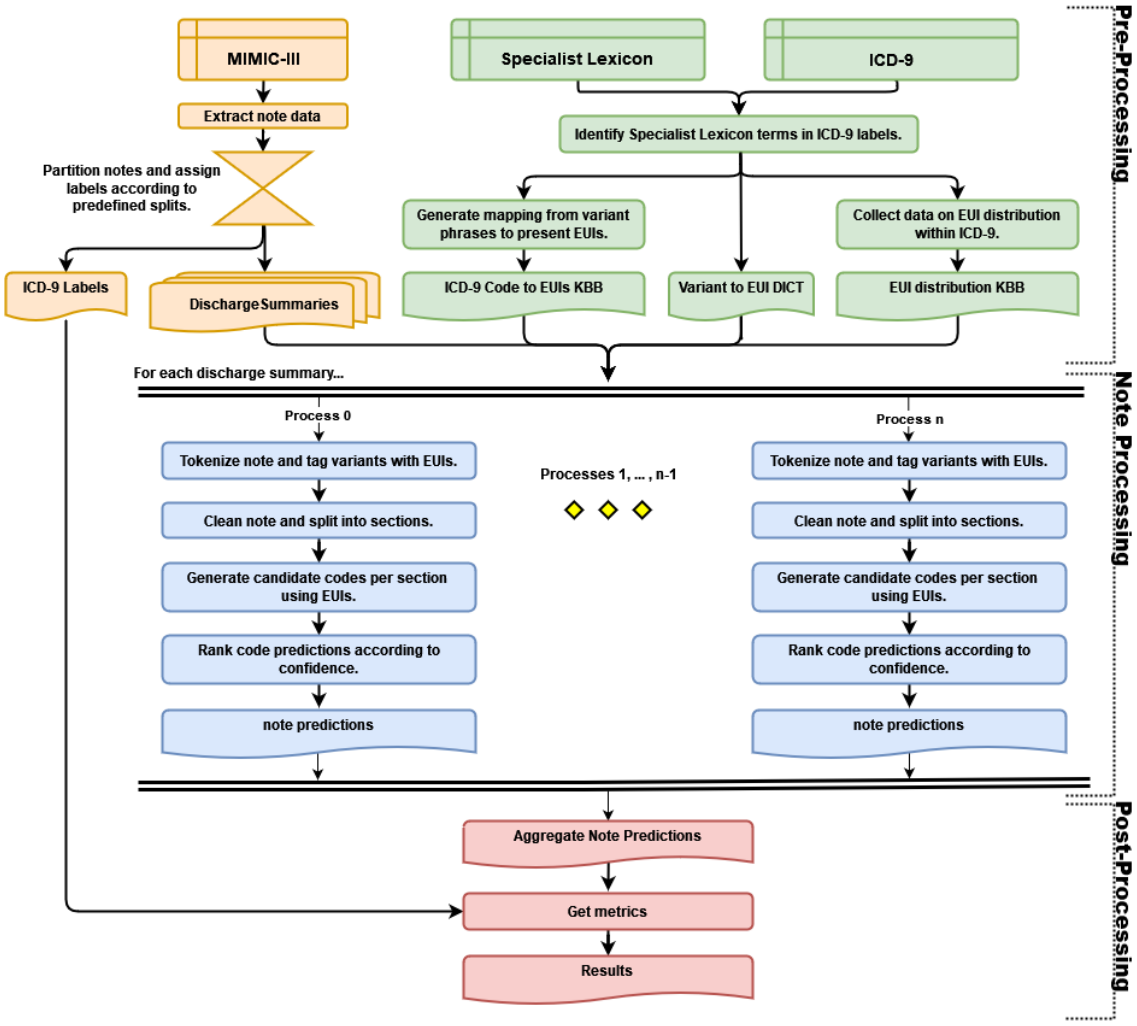

Figure 4.1: Flowchart of our evaluation pipeline on the MIMIC-III coding dataset. In the figure, stages of the pipeline are encoded by color: yellow corresponds to MIMIC dataset pre-processing, green to medical knowledge-base integration, blue to note processing, and red to post-processing.

first three characters for diagnosis codes, with the exception of E codes for which a decimal is added after the first four characters. In cases where the reformatted code ends with a decimal, the decimal is dropped. As a final step in pre-processing the dataset, the resulting table of discharge summaries and ICD-9 code labels is filtered

MIMIC-III NOTEEVENTS DATA TABLE

Column Name	**Description**
ROW_ID	Unique identifier for row index in the table
SUBJECT_ID†	Unique patient identifier
HADM_ID†	Unique hospital admission identifier
CHARTDATE	Date on which the note was charted
CHARTTIME	Time at which the note was charted
STORETIME	Time at which the note was stored
CATEGORY†	Category of the note, e.g. *Discharge summary*
DESCRIPTION†	Additional description of note, e.g. whether the note is a report or addendum
CGID	Unique caregiver identifier
ISERROR†	Whether the note contains an error
TEXT†	The de-identified text of the note

Table 4.1: Column descriptions for the NOTEEVENTS table of MIMIC-III v1.4 [23]. Column names marked † are utilized in our approach.

according to each of the splits: *Full*, *Top 50*, and *Rare 50*. The resulting dataframes are saved to disk for later use and the discharge summaries are written to individual text files indexed by hospital admission ID, for use in the note processing stage.

4.2 Domain Knowledge Integration

The Unified Medical Language System serves two key roles in our system: the first is to help identify clinically significant terms within ICD-9 titles and the second is to resolve ambiguous domain-specific language. To the first end, we utilize the UMLS term mapping utility to normalize terms within ICD-9 titles by mapping them to alphanumeric lexical identifiers in the Specialist Lexicon, known as Entry Unique Identifiers. These terms can be either single words or n-grams within the

title. For, example, the ICD-9 code 285.1, with title *Acute posthemorrhagic anemia*, would generate the EUIs E0007202, *acute posthemmorhagic anemia*; E0049207, *posthemmorhagic*; E0007127, *acute*; and E0008920, *anemia* (see Figure 4.2, Step 1).

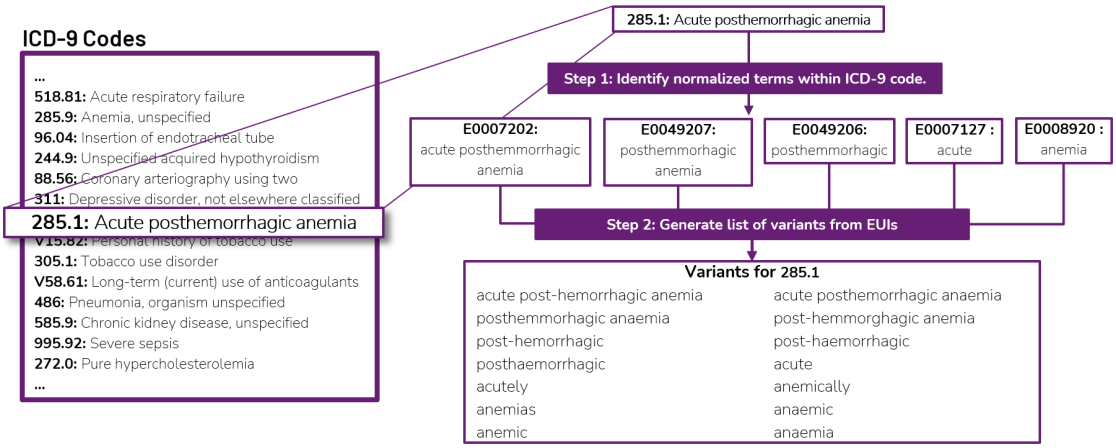

Figure 4.2: Outline of the pipeline for term normalization and variant generation for ICD-9 titles.

In the second step, we leverage the normalized terms identified in the first step to generate sets of alternative forms of these terms, a process which is depicted in Step 2 of figure 4.2. These alternative forms, or *variants*, include at minimum abbreviations, acronyms, plural forms, conjugations, and spelling variations. To accomplish this, we use the Lexical Variant Generation (LVG) command line utility included with the Specialist Lexicon tools [47]. The LVG takes a term or list or terms as input and outputs a list of variants according to the specified flow control options. These options include normalization methods like stripping punctuation and diacritics, and splitting ligatures as well as derivational options like generating fruitful variants, inflections, synonyms, and spelling variants (for a comprehensive list, see Appendix B). Since prior work has shown that the optimal choice of variant types can vary by task [17], we experiment with different variant generation setups. The

first approach, which we dub the *lexical expansion approach*, generates variants using spelling variants, acronyms and abbreviations, and derivations. The second approach, which we dub the *semantic and lexical expansion approach*, generates variants using the fruitful variants flag, which includes the a comprehensive set of variations, including spelling variants, inflections, synonyms, acronyms and abbreviations, expansions of abbreviations and acronyms, and derivations [11].

The objective of this stage is ultimately to integrate domain knowledge into our NLP++ analyzer, so as a last step in the process, we convert the raw data output by the UMLS utilities into NLP++ knowledge bases and dictionaries. The resulting files are organized as follows:

1. Knowledge Bases

   ID to EUI: Maps each variant to an Entry Unique Identifier (EUI) or list of possible EUIs.

   EUI to ICD Code: Maps each EUI from **ID to EUI** knowledge base to a list of ICD codes with titles containing the normalized EUI term.

   ICD Code to EUI: Maps each ICD code in the set to a list of EUIs such that each EUI term is contained within the ICD code title.

2. Dictionaries

   Variant to ID: Maps each lexically unique variant to an unique integer id corresponding to an index in the **ID to EUI** knowledge base.

   Negations: Negation terms with attribute defining pre- or post-negation for use with the NegEx algorithm.

4.3 Note Processing

In the note processing stage, we take a set of notes, in this case each of the test sets of MIMIC-III, and output a set ICD-9 codes for each note ranked by relevance. Our approach involves three steps: extraction, linking and ranking. In the first step we decompose the input text into structured sections. In the second step we extract key terms link these to central concepts. In the third step we rank the set of extracted codes using a inverse-document frequency-based method. In this section we give an overview of the analyzer structure and experimental setup to investigate the effects of knowledge sources and ranking formulations on overall ICD coding performance.

We first describe the general structure of the NLP++ analyzer. Our analyzer consists of 27 distinct passes (for a comprehensive list with pass types, see Appendix A) each of which performs a distinct step in note processing. The first step for NLP++ analyzers is the tokenization step in which we perform tokenization of the input note using the built-in *Dictionary Tokenizer* in NLP++. The Dictionary Tokenizer uses a word-based tokenization strategy which splits the text on whitespace and punctuation. Additionally, strings containing both digits and letters are split into tokens containing either all letters or all numbers. The tokenization pass also performs lookups in the dictionaries for each token in the parse tree. If a token matches an entry in one of the dictionaries, its attributes are added to the corresponding parse tree node, as in Figure 4.3. In the case that a multi-token phrase is matched, the entire token sequence match is reduced to a *_phrase* node in the parse tree 4.3. The output of the tokenization step is thus a shallow parse tree consisting of tokens from the input text which are tagged with negation type and an integer EUI identifier, where applicable.

The next three passes of the analyzer-2: *KBFuncs*; 3: *array_funcs*; and 4:

```
1 [0,0,0,0,0,0,num, ("NOSP" 1)]
. [1,1,1,1,0,0,punct, ("NOSP" 1)]
\_ [2,2,2,2,0,0,white]
No [3,4,3,4,0,0,alpha, ("SP" 1) ("cap" 1) ("negation" "PREN")]
\_ [5,5,5,5,0,0,white]
definite [6,13,6,13,0,0,alpha, ("SP" 1) ("lower" 1)]
\_ [14,14,14,14,0,0,white]
signs [15,19,15,19,0,0,alpha, ("SP" 1) ("lower" 1) ("conid" 56567)]
\_ [20,20,20,20,0,0,white]
of [21,22,21,22,0,0,alpha, ("SP" 1) ("lower" 1) ("conid" 65427)]
\_ [23,23,23,23,0,0,white]
_phrase [24,41,24,41,0,0,node, ("SP" 1) ("lower" 1) ("conid" 41621)]
  vascular [24,31,24,31,0,0,alpha, ("SP" 1) ("lower" 1) ("conid" 6550)]
  \_ [32,32,32,32,0,0,white]
  occlusion [33,41,33,41,0,0,alpha, ("SP" 1) ("lower" 1) ("conid" 7242)]
. [42,42,42,42,0,0,punct, ("NOSP" 1)]
```

Figure 4.3: A section of the parse tree for a note after the tokenization pass. Terms from the NegEx dictionary are tagged with the 'negation' attribute. Terms with entries in the Specialist Lexicon are tagged with 'conid' integer inidices and multi-word terms are reduced to *_phrase* nodes. Spaces in the input text are represented by the '_' token in the parse tree.

pn_funcs-are declarative passes which define functions which are called throughout the analyzer. KBFuncs is a built-in pass, or library pass, that provides useful functions for working with knowledge bases. In particular, we utilize the *AddUniqueStr*, *AddUniqueConVal*, and *AddUniqueCon* functions to add unique strings, concepts, and values to knowledge bases, respectively, along with functions which facilitate exporting knowledge bases. In the *array_funcs* pass, we define functions to perform common array operations, including array concatenation, element swapping, QuickSort, binary search, duplicate filtering, and conversion functions for interoperability with knowledge base data structures. The *pn_funcs* pass includes a single function–*PnrPushUniqueVal*–which appends a value to a parse tree node's variable.

Passes 6 through 19 A perform cleaning of the note text and organization of the parse tree. Starting with pass 6, *clean_notes*, we excise non-relevant information including de-identified placeholder strings of the form '[** <previously-identifiable text >**]' as well as headers and footers, which empirical investigation suggests predominantly contain metadata. After filtering, we begin to organize the parse tree into structural components, in order of increasing granularity: sections, subsections, enumerated lists, and sentences (see Figure 4.4 for an example of the resulting parse tree structure). For sections and subsections, header names, 'History of Patient Illness' or 'Chief Complaint' for example, are added as attributes on the parent node when present. Finally, we clean all whitespace from the parse tree, including space characters, tabs, and newlines.

Pass 20, *gather_negations* implements the NegEx algorithm with a maximum distance of 5 nodes between a negation term and a clinical entity. Our negation window size follows the original NegEx implementation [7] to its relative effectiveness and ease of implementation, though some work has shown improvement using a dynamic window size [31]. The first rule in the pass matches a leaf node tagged as *pre*-negation,

Figure 4.4: Beginning of parse tree after pass 19: *remove_whitespace*. *_section* nodes beneath the root node have been collapsed to demonstrate the hierarchical structure of the parse tree, with the exception of the *Social History* section. Each *_sentence* node in the social history section contains the tokenized text of a single sentence from the section. The 'fired' tag refers to the fact that the whitespace cleaning rule in pass 19 has been applied to the node.

along with the next 5 sibling nodes or up to the next *_section*/*_subsection*/*_sentence* boundary, whichever comes first. Since compound medical terms are reduced to a single *_phrase* node, they are treated as a single entity, or node match. The following example outlines the steps of the algorithm implementation:

1. Chest radiography showed $\underset{\text{pre-negation}}{\underline{\text{no signs of}}}$ $\underset{\text{EUI}}{\underline{\text{pleural effusion}}}$ or $\underset{\text{EUI}}{\underline{\text{pneumothorax}}}$.
2. Chest radiography showed <pre-negation >pleural effusion or pneumothorax.
3. Chest radiography showed <pre-negation ><EUI ... >or <EUI ... >.
4. Chest radiography showed <pre-negation >$\underset{1}{\underline{\text{EUI}}}$ $\underset{2}{\underline{\text{or}}}$ $\underset{3}{\underline{\text{EUI}}}$.
5. Chest radiography showed <negated >

Figure 4.5: Parse tree after the completion of Pass 21. Normalized term identifiers are shifted to parent nodes under the attribute *conid*.

The next rule performs the same operation for *post*-negation terms, instead excising the preceding 5 nodes.

Pass 21-27 A perform the term extraction and ranking steps. The aim of these steps is to take the structured parse tree with terms tagged for normalization and rank the importance of the terms using a term-frequency inverse-document-frequency based method [48] with the set of all ICD-9 titles as the reference corpus. We start by copying all tagged terms in the parse tree onto parent nodes, so that each *section*, *subsection*, and *sentence* node contains a list of all normalized terms (represented by unique identifiers) contained within.

For any given ICD-9 code $c \in C$, we represent it as a set of terms that occur within its title such that $T_c = \{t_1, t_2, ..., t_n\}$ (for an example, see Figure 4.2). We then calculate the frequency of each unique term within all ICD-9 titles, given by f_t, to encode the relative specificity of each term (lower corpus frequency => higher specificity) [48]. We then define the total weight of a code, $\mathcal{W}_c$, as the sum of the

inverse document frequencies (IDFs) of each of its constituent terms, $t \in T_c$ over the ICD-9 corpus, C, as follows:

$$\mathcal{W}_t = \frac{f_t}{|C|} \tag{4.1}$$

$$\mathcal{W}_c = \sum_{i}^{|T_c|} \mathcal{W}_{t_i} \tag{4.2}$$

We then use the same IDF term weights to calculate the ranking score of a code with respect to a particular note. Let $G = t_1, t_2, ..., t_m$ be the set of all terms in the document of interest and $H = G \cap T_c$ be the set of codes in both the input text and the code c, then the rank of code c with respect to the note, $\mathcal{R}_c$, is given by the following:

$$\mathcal{R}_c = \frac{\sum_{j=1}^{|H|} \mathcal{W}_{H_j}}{\mathcal{W}_c} \tag{4.3}$$

By dividing by the total possible code weight, we ensure that the ranking score for a code is not dependent on the number of terms within its title. Note that unlike TF-IDF, we are not taking into account the frequency of a term within the note.

Since we are ranking a code based on the occurrence of its constituent terms within the target text, we hypothesize that constraining term matches to smaller sections of the text will lead to better performance. To test this we re-formulate our ranking function by assigning a weight to each code for each section s and aggregate the ranking score for each code by applying an aggregation function: *max*, *mean*, or *sum*. We experiment with ranking codes at the *section* and *sentence* level.

Due to the considerable size of the MIMIC-III corpus and the lengthy run-

time of our analyzer, the note processing stage is conducted in parallel on Clemson's Palmetto Cluster. For resources, we allocate 18 nodes with 20 CPUs and 120GB's of memory per node. Notes are first written to individual text files, which are then mapped to available processes with GNU Parallel [51]. Each process runs an instance of the note processing analyzer 4.3 in the NLP++ engine. The final pass in the analyzer, Pass 27, writes the analyzer results to a single-line csv file containing the hospital admission ID (HADM ID) for the discharge summary followed by ranking scores for all ICD-9 codes, in predetermined order. Each of these output files is read and appended to a single csv file which is indexed by HADM ID and has columns corresponding to ranking scores for each ICD code. This final step is also performed in parallel using GNU Parallel to coordinate the process.

4.4 Evaluation

We evaluate all approaches on the test sets of the *Full*, *Top 50* and *Rare 50* splits of MIMIC-III. Following previous work [33, 54], we use the receiver operating charateristic area under the curve ($ROC\ AUC$), F1-score. For the top set, we additionally use precision at k, for $k = 5$. Since ICD-coding is a multi-class classification problem, we provide ROC AUC and F1-score results using both *macro*- and *micro*-averaging. Macro-averaging uses the arithmetic mean of the metric for each individual class, thus weighting each class equally. Micro-averaging, on the other hand, aggregates the results of all classes to compute the average metric, giving equal weight to each instance. We give a brief explanation of each of these metrics.

The receiver operating characteristic (ROC) curve was originally developed in World War II to gauge the ability of radar operators to distinguish a signal (e.g. aircraft or warships) from noise [13]. In modern practice, the ROC curve is widely

used to test the performance of a classifier. The ROC curve plots sensitivity, or True

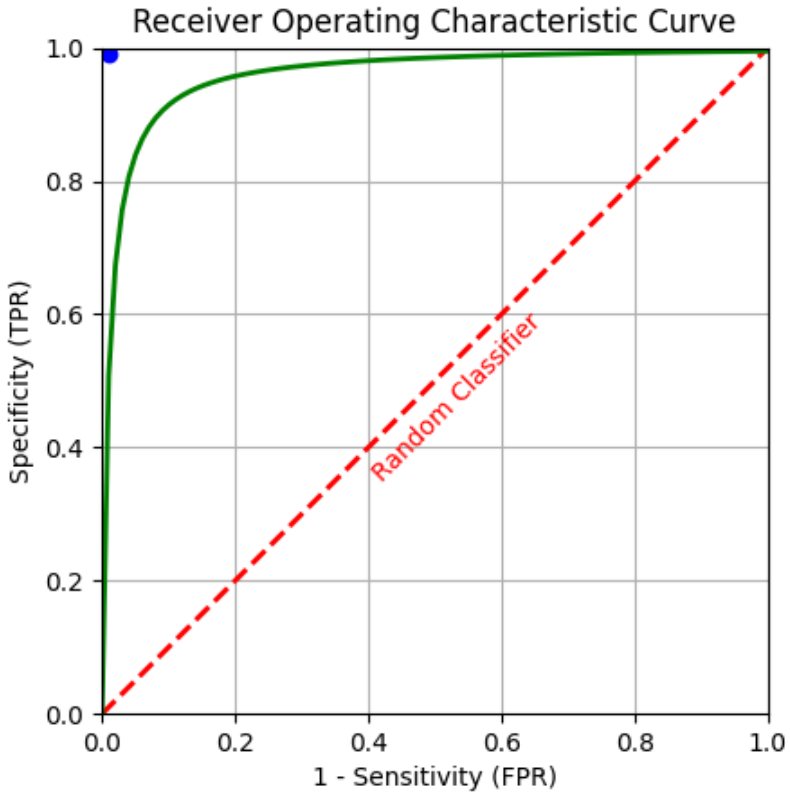

Figure 4.6: Example receiver operating characteristic curve. The red line represents an ROC area under the curve of 0.5, or a random classifier, the blue point at $(0.0, 1.0)$ represents a perfect classifier, and the green curve represents a relatively good classifier.

Positive Rate, versus $1-$specificity, or False Positive Rate, across all threshold values (see Figure 4.6). The Area Under the Curve (AUC) of the ROC thus provides a scalar value between 0.0 and 1.0 that represents the general performance of a classifier for all threshold values.

The F-score, or F-measure, is the weighted harmonic mean of recall (R) and precision (P) which is often expressed in terms of the parameter β that weights the

relative importance of recall over precision (Equation 4.4) [45].

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{4.4}$$

$$F_1 = \frac{2PR}{P + R} \tag{4.5}$$

We evaluate our system using F-score with a β of 1, referred to as F1-score (4.5), so that precision and recall are weighted equally.

Precision at k is a metric which is frequently used to measure the relevance of the top results of an information retrieval system [45] and is defined as the precision using only the k highest-ranked labels [33]. Since the mean number of labels for the Full set of MIMIC-III is approximately 15, k-values of 5, 8 and 15 are frequently used in the literature [33, 54]. Since the mean number of labels for the top-50 set is 5.7, we report precision at k with $k = 5$. Likewise, since the mean number of labels in the Rare set of MIMIC-III is approximately 1, results for this metric are not reported.

Chapter 5

Results

5.1 Overview of Results

In this chapter we provide results of our experiments and some discussion of these results. We present results separately for each of the subsets of MIMIC-III on which we test: *top-50* and *rare-50*. We also provide a comparison to previous state-of-the-art methods, including CAML [33], MultiResCNN [28], LAAT [53], PLM-ICD [19], and KEPTLongformer [54] for the top-50 set and KEPTLongformer and MSMN [55] for the rare-50 set. We select methods that were state-of-the-art at the time of publication and had published results for the target dataset.

We denote each of our methods as follows: *LexSyn* refers to the use of lexical variants and synonyms for normalization. The subscript refers to the aggregation method-*max*, *sum*, or *mean*-and the scope of term matches-*sent* for sentence-level, *sect* for section level, and *full* for the full note. Results for the complete MIMIC-III test set are available in Appendix C for space reasons.

MIMIC-III Rare 50 Results

		ROC AUC		F1-Score	
	Approach	Micro	Macro	Micro	Macro
Previous	$\text{MSMN}_{\text{pretrained}}$	76.2	75.3	17.1	17.2
	$\text{MSMN}_{\text{zero-shot}}$	48.9	52.3	3.5	4.0
	$\text{MSMN}_{\text{finetuned}}$	44.0	58.2	3.3	4.2
	$\text{KEPTLongformer}_{\text{pretrained}}$	82.3	81.4	30.9	25.8
	$\text{KEPTLongformer}_{\text{zero-shot}}$	76.5	74.9	16.7	15.2
	$\text{KEPTLongformer}_{\text{finetuned}}$	83.3	82.7	32.6	30.4
Our Results	$\text{LexSyn-Section}_{\text{max}}$	**77.8**	80.0	**12.6**	24.7
	$\text{LexSyn-Section}_{\text{mean}}$	76.7	77.2	12.4	23.4
	$\text{LexSyn-Section}_{\text{sum}}$	77.0	80.4	12.5	20.8
	$\text{LexSyn-Sentence}_{\text{max}}$	76.2	**81.0**	10.2	28.2
	$\text{LexSyn-Sentence}_{\text{mean}}$	74.0	77.6	8.8	28.2
	$\text{LexSyn-Sentence}_{\text{sum}}$	75.9	80.8	10.3	**29.2**
	LexSyn-Full	**77.8**	80.0	12.2	23.6

Table 5.1: Results on the MIMIC-III Rare 50 test set. Best-performing scores for each of our approaches are bolded. Results for previous approaches are taken from Yang et al., 2022 [54].

5.1.1 Rare 50

Results for the *rare-50* split are displayed in Figure 5.1.1. We compare our results to the two state-of-the-art models evaluated by the original authors: KEPT-Longformer [54] and MSMN [55]. Each of these models is evaluated in three different training setups: *pretrained*, *zero-shot*, and *finetuned*. Pretrained refers to the setup in which the pretrained language model is directly finetuned on the rare-50 train set before evaluation on the test set. Zero-shot refers to the setup in which a model is trained on the top-50 test set and then directly evaluated on the rare-50 set. Fine-tuned refers to the setup in which the model is first finetuned on the top-50 set then further trained on the rare-50 train set before evaluation on the rare-50 test set. This last setup also includes training with an additional Hierarchical Self-Alignment

Pretraining objective in order to learn an embedding space in which similar medical terms are grouped closer together.

We find that our LexSyn-Section$_{\max}$ analyzer achieves a level of performance comparable to recent state-of-the-art approaches in terms of ROC AUC, falling within 4 points of the best-performing deep-learning model, KEPTLongformer$_{\textit{finetuned}}$. Despite the slight performance difference, we identify a few key advantages which support the utility of our analyzer in a clinical setting. The first of these is explainability: our system is fully traceable and provides evidence from the text to support a particular code assignment. Furthermore, our approach does not require any training data (labeled or unlabeled) which is advantageous in a low-resource setting.

5.1.2 Top 50

Results for the *top-50* split are displayed in Figure 5.1.2. We find that on the $top-50$ split, our results fall short of recent state-of-the-art deep learning methods in all evaluation metrics. We suspect that this is due in part to the *top-50* split being an easier task for deep-learning based methods. In particular, the smaller label space (50 labels vs 8,922 for the full set) and the large number of samples per label make this dataset significantly less challenging for deep-learning models than both the *rare-50* and *full* sets. We nonetheless find that our approaches achieve a reasonable performance baseline and identify a few strategic modifications that could potentially increase performance. We find that our sources of error on this dataset fall into four main classes: partitioning, numeric or tabular data, semantic linking, and idiopathic issues.

We observe that performance for individual codes on the Top-50 dataset is highly variable as evidenced by the per-code F1-scores in Figure 5.1. Using our

MIMIC-III Top 50 Results

		ROC AUC		F1-Score		Prec. @ k
	Approach	Micro	Macro	Micro	Macro	5
Previous	CAML [33]	91.1	87.5	52.4	60.6	61.1
	MultiResCNN [28]	92.4	89.7	62.2	67.3	63.4
	LAAT [53]	92.8	90.5	60.8	66.8	64.0
	PLM-ICD [19]	93.8	91.7	66.3	70.5	65.7
	KEPTLongformer [54]	94.8	92.6	72.9	68.9	67.3
Our Results	LexSyn-Section$_{\text{max}}$	69.8	70.9	33.3	37.1	29.4
	LexSyn-Section$_{\text{mean}}$	67.8	68.9	30.1	33.7	26.3
	LexSyn-Section$_{\text{sum}}$	68.6	71.3	31.7	39.6	27.6
	LexSyn-Sentence$_{\text{max}}$	69.6	69.1	31.3	34.4	29.0
	LexSyn-Sentence$_{\text{mean}}$	**71.7**	**72.2**	**34.5**	37.1	**31.8**
	LexSyn-Sentence$_{\text{sum}}$	70.0	**72.2**	32.9	**40.8**	27.7
	LexSyn-Full	68.0	68.0	31.9	38.5	29.6

Table 5.2: Results on the MIMIC-III Top 50 test set [33]. LexSyn denotes the normalization type: lexical and semantic normalization. Section, subsection, and sentence subscripts refer to the scope to which we assign codes.

LexSyn-Section$_{\text{max}}$ analyzer as a baseline, we conduct error analysis for the five ICD-9 codes with the worst individual F1-scores, listed in Table 5.3. We first plot the confusion matrices, seen in Figure 5.2, for each of these codes to better understand where our analyzer is failing. We observe that for two of the three codes with an F1 of 0.0–codes *285.1* and *39.61*–a positive label is never predicted. In general, we find that our analyzer tends produce more false negatives for these codes than false positives. The exception to this trend is the code *410.71*, which has the opposite problem: a much higher rate of false positives than false negatives. To identify the source of these problems, we manually review a selection of notes from the test set.

We struggle to identify a consistent source of error for code *412*, Old myocardial infarction, though we do note the challenge of distinguishing between this code and, for example, code *410*, Acute myocardial infarction, or even *410.71*, Subendocardial

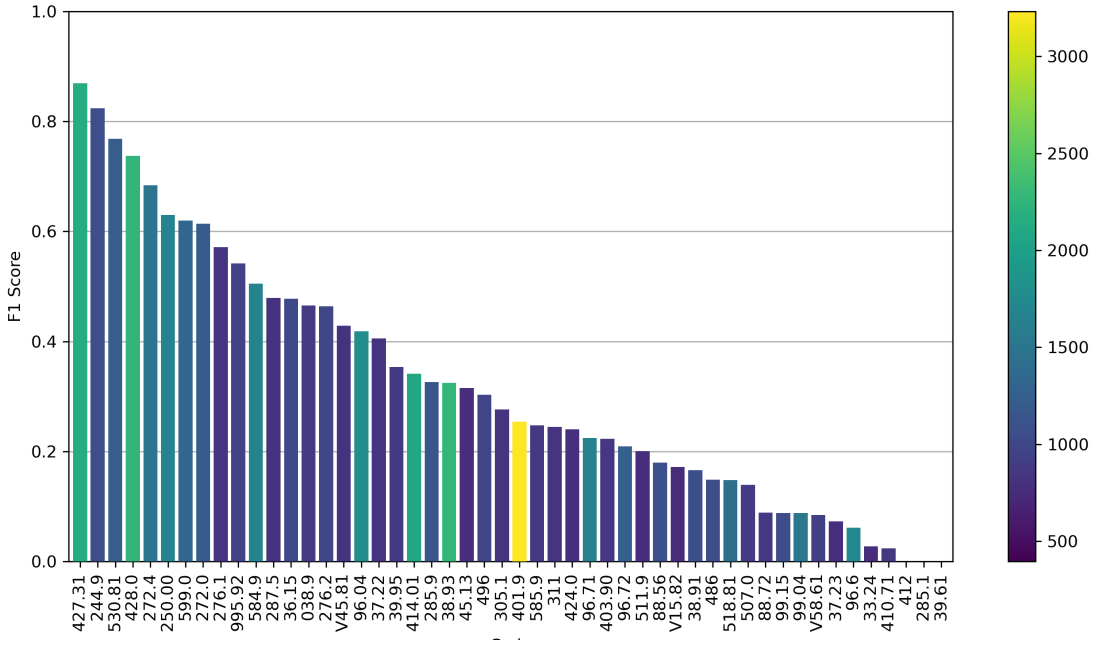

Figure 5.1: F1-score per code on the Top-50 dataset, sorted by decreasing F1-score. Bar colors represent frequency of the code in the Top-50 training set; the color scale is displayed on the bar to the right of the plot.

Code	**Title**	**Freq.**	**F1**
412	Old myocardial infarction	121	0.0
285.1	Acute posthemorrhagic anemia	203	0.0
39.61	Extracorporeal circulation auxiliary to open heart surgery	226	0.0
410.71	Subendocardial infarction, initial episode of care	91	0.02
33.24	Closed [endoscopic] biopsy of bronchus	127	0.03

Table 5.3: Test set frequencies, F1-scores and titles for the five worst-performing codes on the Top-50 set using the LexSyn-Section$_{\max}$ analyzer.

infarction, initial episode of care. We find that in the notes we review, Old myocardial infarction is almost exclusively assigned when "myocardial infarction", or its initialism MI, occurs in the *Past Medical History* section of the discharge summary. In fact, applying a simple matching rule for these terms in the *Past Medical History* section

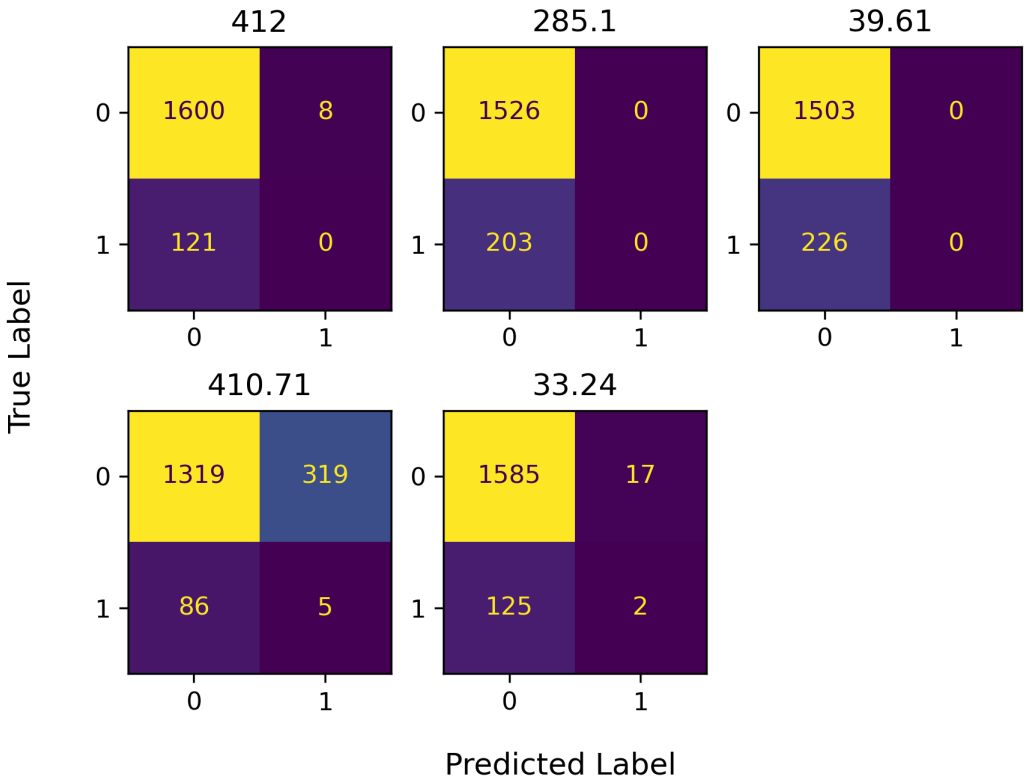

Figure 5.2: Confusion matrices for the worst-performing codes on the top-50 set in terms of F1-Score.

significantly outperforms our approach, with an F1 score on the Top 50 test set of 58.2.

In the discharge summaries that we review for code *285.1*, Acute posthemorrhagic anemia, we find that the terms themselves do not appear in the text. We suspect that the indicator of this code comes from blood sample results, for which abnormal red blood cell counts and hemoglobin levels are marked by an asterisk. This type of inference from non-textual signifiers or numerical data is outside the scope of our analyzer, though one could add a heuristic rule to help identify these cases.

For code *39.61*, Extracorporeal circulation auxiliary to open heart surgery, we

find that the title and key subphrases of the title (e.g. *extracorporeal circulation*) do not occur as such in the text. In the set of discharge summaries selected for review, we observe the presence of procedures which may classify as extracorporeal circulation methods, for example "CPB", an initialism for cardiopulmonary bypass. Further investigation reveals that cardiopulmonary bypass (UMLS CUI: C0007202) is defined as a *narrower*, or child, concept of extracorporeal circulation (UMLS CUI: C0015354). This suggests that leveraging ontological information beyond just synonyms may be necessary to improve performance. We note a similar effect for code *33.24*, Closed [endoscopic] biopsy of bronchus, for which the positive signal in the note often appears to be a specific procedure term, like *bronchoscopy*.

We are unable to identify a consistent source of error for the code *410.71*, Subendocardial infarction, initial episode of care. We do, however, note the relative paucity of the term subendocardial in the Top 50 test set, which occurs in only two notes. We suspect that the high count of false positives produced by our analyzer stems from the frequency of the term infarction in notes in the test set coupled with its relative infrequency in ICD-9 titles.

In our error analysis of the Top-50 set, we find that our analyzer does not effectively leverage code frequency within individual notes. As described in Section 4.3, our analyzer generates code ranks for each sentence or section (with the exception of the full note analyzer). This quite often results in a large number of ranking scores for codes in a single note. To handle the situation in which a code is assigned a rank multiple times in the same note, we use one of three aggregation methods: the mean of the ranks, the sum, or the median. While both sum and median aggregation leverage the frequency of code appearance to some extent, we suspect this could also adversely affect performance due to the incorporation of noisy ranking scores. To test this hypothesis, we add a filtering step to the raw output of the analyzer and

re-calculate results. To filter ranking scores for a note, we calculate the mode of the scores generated by our analyzer and use this as a threshold (see Figure 5.3). Scores below this threshold are set to zero and scores above the mode remain unchanged. After filtering, we aggregate the results using the three aggregation methods described in Section 4.3 and evaluate on the test set. Results are displayed in Table 5.1.2. For the sum aggregation method, we find that filtering increases micro-averaged ROC AUC by 1.2 points and macro-averaged ROC-AUC by 2.2 points, resulting in our overall best-performing analyzer for the Top-50 test set in terms of macro-ROC AUC. On the other hand, we find that filtering has an adverse effect when applying mean aggregation and likewise negatively impacts performance on the rare set.

	ROC AUC		F1-Score	
Approach	Micro	Macro	Micro	Macro
LexSyn-Sentence$_{\text{mean}}$	**71.7**	72.2	**34.5**	37.1
LexSyn-Sentence$_{\text{sum}}$	70.0	72.2	32.9	**40.8**
Filtered-Sentence$_{\text{mean}}$	71.1	**74.4**	32.9	30.8
Filtered-Sentence$_{\text{sum}}$	70.2	70.2	31.3	28.5

Table 5.4: Results on the Top 50 test set after filtering ranking scores below the mode. Original results are on the top two rows and results with filtering are on the bottom two rows.

One of the challenges of the Top-50 test set is the variability in the number of ground truth labels per note. Choice of activation threshold value when converting ranking scores to binary predictions can thus significantly impact overall F1-score as demonstrated in Figure 5.4. We test the effect of different threshold values on the test set and calculate resulting F1 scores. We find that activating scores above the mean score per note provides a reasonably good choice of threshold.

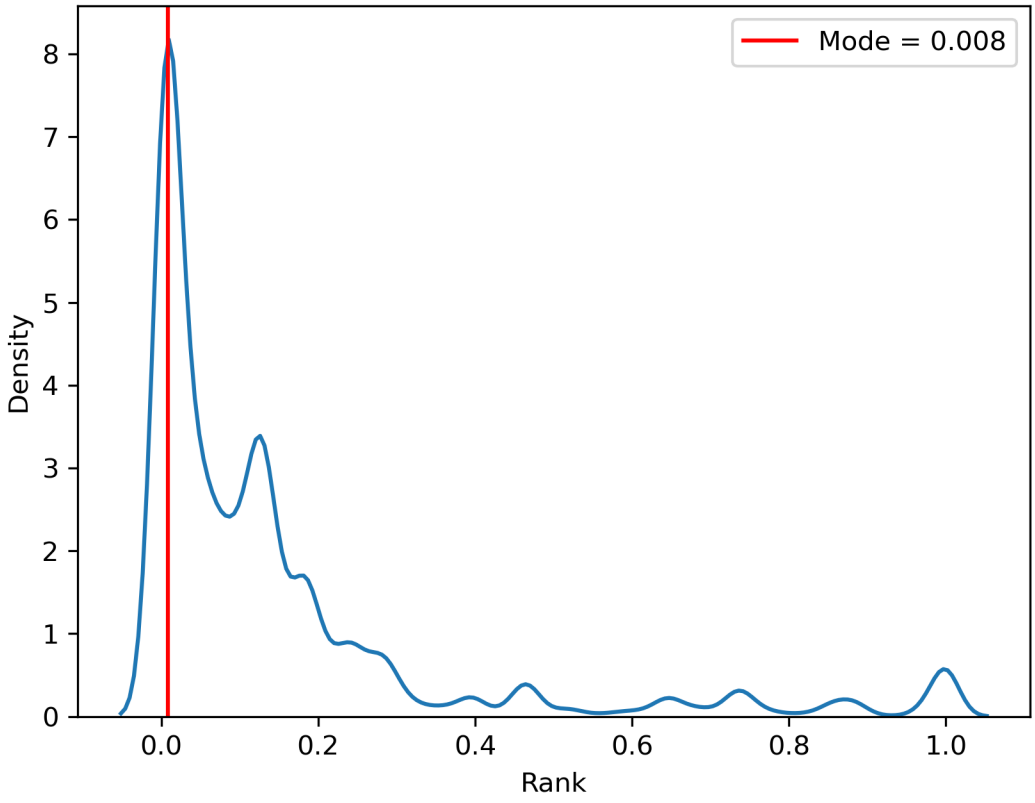

Figure 5.3: Kernel Density Estimation plot using a Gaussian Kernel for per-sentence code ranks on the Top-50 test set. We find the mode of all per-sentence code ranks to be an effective threshold for filtering low-quality rank assignments.

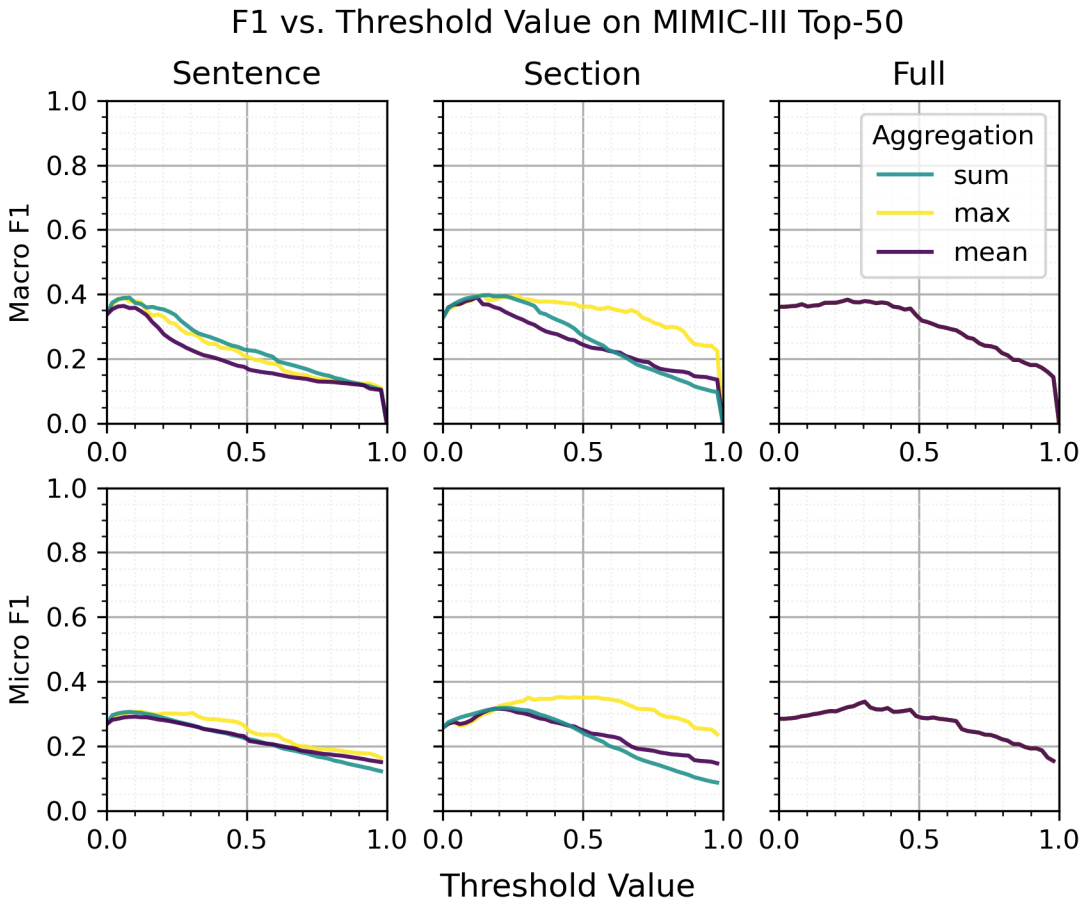

Figure 5.4: To produce a final set of codes for evaluation from the ranking scores assigned by our analyzer, a threshold value is chosen per note and codes with ranking scores above this value are added to the set. Here we plot of micro- and micro-averaged F1-score vs choice of threshold value for each scope and aggregation method on the MIMIC-III Top-50 test set.

Chapter 6

Future Work

6.1 Word Sense Disambiguation

Our system tends to have higher recall than precision. This could partially be caused by our approach to extracting clinical entities which does not differentiate between semantic interpretations of a particular medical entity. This is particularly salient for abbreviations and acronyms, which often require contextual clues to disambiguate [43]. Consider, for example, the term 'ms', which maps to 12 unique concepts in the 2007AC UMLS [43]: *Marinesco-Sjogren syndrome*, *metric system*, *Mississippi (geographic location)*, *mitral valve stenosis*, *Montserrat*, *multiple sclerosis*, *milliseconds*, *MTR gene*, *academic degree (Master of Science)*, *supernumerary mandibular left primary canine*, *microbiology susceptibility domain*, and *multiple sclerosis (susceptibility to)*. Our system would, in practice, give equal weight to each of these *senses* of the term 'ms' without attempting to identify the true sense of the term in the text. An lucrative path for future work may be to incorporate heuristic algorithms for word-sense disambiguation (WSD) [44, 8] into the entity extraction passes of the analyzer.

6.2 Leveraging Ontological Information

In our system we utilize the Lexical Variant Generator of the UMLS to identify variants of key medical terms. This allows us to normalize these variants in the text by mapping them to a central concept. We experiment with different variant generation setups as outlined in Section 4.2. With the exception of a couple of disparate studies [11, 17], we find the literature on lexical normalization for medical entities to be lacking. In particular, an in-depth analysis of the downstream impact of different variant generation setups would be a useful tool for guiding the construction of systems that utilize the lexical tools. Additionally, we hypothesize that our system could better leverage existing ontological information, including free text descriptions and hierarchical relationships.

6.3 Data Quality

The dataset on which we evaluate our system, MIMIC-III [23], is a large and meticulously compiled dataset which has realized significant progress toward the systematic study of medical coding. Due to the increasing ubiquity of MIMIC in the literature on medical coding, the validation of ground truth labels in the dataset has become supremely important [46]. Perhaps the most critical area for future research that we identify is the development of gold-standard labels for the MIMIC dataset. On a broader scale, the impact of imperfect–or silver-standard–labels is pernicious, as it limits the ability to empirically compare different approaches and could falsely conflate (or deflate) results. The lack of open-access, quality coding datasets also inhibits the ability of researchers to gauge the capacity of a system to generalize across domains. It may be the case that notes or label-assignments

within the MIMIC dataset, having been sourced from a single hospital system [23], are measurably different from those originating from other healthcare providers. This hypothetical difference could result in significant performance degradation for many state-of-the-art systems. However due to the paucity of other openly available coding datasets, this hypothesis is difficult to test without a considerable annotation budget and access to expert annotators.

6.4 Explainability

To better investigate the explainability of our system, we build a visualization tool for clinical notes in HTML. Our visualization tool is adapted from the entity resolver visualizer from Spark NLP [24] which places a color-coded bounding box with corresponding labels around target entities in the text, as seen in Figure 6.1. While the Spark NLP visualizer is a useful tool, there were a few drawbacks that

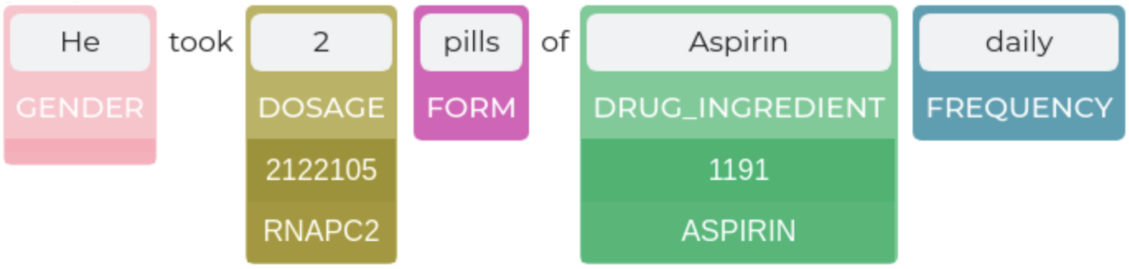

Figure 6.1: An example visualization of the Spark NLP visualizer applied out-of-the-box [20].

limited its application to our use-case. The first of these is the size and frequency of the bounding boxes. The high frequency of tagged entities in our documents and the length of text in our tags inhibit legibility, often limiting the resulting document visualization to just a few tagged terms per line. To address this issue, we replace

the static bounding boxes with color-coded highlighting in the text and add dynamic information boxes which appear on hover. We color code terms in the text by the ICD-9 code with which the term is associated. In the hover box we add the ICD-9 code, its title, and the ranking score assigned by the analyzer. We also add a header containing color references for each of the ICD-9 codes identified in the text. While we find this to be a useful tool for tracing the classification process of our analyzer, more work is needed to turn this into a tool that can be deployed in practice.

Appendices

Appendix A Analyzer Pass Structure

#	Pass	Type
1	dicttokz	Tokenizer
2	KBFuncs	@DECL
3	array_funcs	@DECL
4	pn_funcs	@DECL
5	init_kb	@CODE
6	clean_notes	@RULES
7	get_negation	@RULES
8	get_breaks	@RULES
9	get_sections	@RULES
10	get_loose_passages	@RULES
11	group_loose_passages	@RULES
12	remove_breaks	@RULES
13	get_subsection_headers	@RULES
14	get_subsections	@RULES
15	get_list_items^{R}	@RULES
16	get_lists	@RULES
17	get_sentences	@RULES
18	sentences	@RULES
19	remove_whitespace	@RULES
20	gather_negations	@RULES
21	shift_keywords	@RULES
22	keyword_funcs	@DECL
23	set_line_count	@CODE
24	extract_codes	@RULES
25	rank_codes	@CODE
26	aggregate_and_predict	@CODE
27	kb_out	@CODE

Table 1: Pass structure in the NLP++ ICD-coding analyzer for MIMIC-III notes. Passes marked ^{R} are recursive.

Appendix B Lexical Variant Generator Flow Components

```
-f:0        Strip NEC and NOS.
-f:a        Generate known acronym expansions.
-f:A        Generate known acronyms.
-f:An       Generate antiNorm.
-f:b        Uninflect the input term.
-f:B        Uninflect words.
-f:Bn       Normalized Uninflect words.
-f:c        Tokenize.
-f:ca       Tokenize keep all.
-f:ch       Tokenize no hyphens.
-f:C        Canonicalize.
-f:Ct       Lexical name.
-f:d        Generate derivational variants.
-f:dc~LONG Generate derivational variants, specifying output
 categories
-f:e        Retrieve uninflected spelling variants.
-f:E        Retrieve Eui.
-f:f        Filter output.
-f:fa       Filter out acronyms and abbreviations.
-f:fp       Filter out proper nouns.
-f:g        Remove Genitive.
-f:G        Generate fruitful variants.
-f:Ge       Fruitful variants, enhanced.
-f:Gn       Generate known fruitful variants.
-f:h        Help menu for flow components (this is it).
-f:i        Generate inflectional variants.
```

```
-f:ici~LONG+LONG  Generate inflectional variants, specifying
 output categories and inflections
-f:is      Generate inflectional variants (simple infl).
-f:l       Lowercase the input.
-f:L       Retrieve category and inflection.
-f:Ln      Retrieve category and inflection from database.
-f:Lp      Retrieve category and inflection for all terms begins
 with the given word.
-f:m       Metaphone.
-f:n       No operation.
-f:nom     Retrieve nominalizations.
-f:N       Normalize.
-f:N3      LuiNormalize.
-f:o       Replace punctuation with space.
-f:p       Strip punctuation.
-f:P       Strip punctuation, enhanced.
-f:q       Strip diacritics.
-f:q0      Map symbols to ASCII.
-f:q1      Map Unicode to ASCII.
-f:q2      Split ligatures.
-f:q3      Get Unicode names.
-f:q4      Get Unicode synonyms.
-f:q5      Norm Unicode to ASCII.
-f:q6      Norm Unicode to ASCII with synonym option.
-f:q7      Unicode core norm.
-f:q8      Strip or map Unicode.
-f:r       Recursive synonyms.
-f:rs      Remove (s), (es), (ies).
-f:R       Recursive derivations.
-f:s       Generate spelling variants.
-f:S       Syntactic uninvert.
```

```
-f:Si       Simple inflections.
-f:t        Strip stop words.
-f:T        Strip ambiguity tags.
-f:u        Uninvert phrase around commas.
-f:U        Convert output.
-f:v        Generate fruitful variants from database.
-f:w        Sort by word order.
-f:ws~INT   Word size filter.
-f:y        Generate synonyms.
-f:z        Generate antonyms.
-f:zs       Antonym Substitutions.
```

Listing 1: Flow control options of the Lexical Variant Generator (version lvg.2023) [39].

Appendix C MIMIC-III Full Results

MIMIC-III Full Results

		ROC AUC		F1-Score		Prec. @ k
	Approach	Micro	Macro	Micro	Macro	5
Previous	CAML	98.6	89.5	53.9	8.8	-
	MultiResCNN	91.0	98.6	55.2	8.5	-
	LAAT	98.8	91.9	57.5	9.9	81.3
	PLM-ICD	98.9	92.6	59.8	10.4	84.4
	KEPTLongformer	96.5	85.7	59.9	11.8	84.8
Ours	LexSyn-Section$_{\text{max}}$	62.7	68.0	1.0	2.4	2.3
	LexSyn-Section$_{\text{mean}}$	62.9	69.3	1.1	2.6	1.4
	LexSyn-Section$_{\text{sum}}$	62.6	68.0	1.1	2.7	1.7

Table 2: Results on the MIMIC-III Full test set [33]. Results from previous approaches come from Huang et al.[19]

Bibliography

[1] 104th Congress. Health insurance portability and accountability act of 1996, 1996.

[2] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[3] Randolph C Barrows Jr, M Busuioc, and Carol Friedman. Limited parsing of notational text visit notes: ad-hoc vs. nlp approaches. In *Proceedings of the AMIA Symposium*, page 51. American Medical Informatics Association, 2000.

[4] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl_1):D267–D270, 01 2004.

[5] Allen C Browne, Alexa T McCray, and Suresh Srinivasan. The specialist lexicon. *National Library of Medicine Technical Reports*, pages 18–21, 2000.

[6] J. W. Bush, Leslie Sandridge, Cierra Treadway, Kimberly Vance, and Alberto Coustasse Dr. Ph. Medicare fraud, waste and abuse. 2017.

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

[8] Rachel Chasin, Anna Rumshisky, Ozlem Uzuner, and Peter Szolovits. Word sense disambiguation in the clinical domain: a comparison of knowledge-rich and knowledge-poor unsupervised methods. *Journal of the American Medical Informatics Association*, 21(5):842–849, 2014.

[9] Suhao Chen, Tuan-Dung Le, Thanh Thieu, Zhuqi Miao, Phuong D Nguyen, and Andrew Gin. Computer-assisted medical billing information extraction: comparing rule-based and end-to-end transfer learning approaches. In *2021 IEEE/ACM*

Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), pages 132–133. IEEE, 2021.

[10] Paul Deane, David de Hilster, and Amnon Meyers. Text processing in an integrated development environment (ide): Integrating natural language procesing (nlp) techniques. *PC AI*, 15(5):36–40, 2001.

[11] Guy Divita, Qing T Zeng, Adi V Gundlapalli, Scott Duvall, Jonathan Nebeker, and Matthew H Samore. Sophia: a expedient umls concept extraction annotator. In *AMIA Annual Symposium Proceedings*, volume 2014, page 467. American Medical Informatics Association, 2014.

[12] Hang Dong, Matúš Falis, William Whiteley, Beatrice Alex, Joshua Matterson, Shaoxiong Ji, Jiaoyan Chen, and Honghan Wu. Automated clinical coding: what, why, and where we are? *NPJ digital medicine*, 5(1):159, 2022.

[13] Jerome Fan, Suneel Upadhye, and Andrew Worster. Understanding receiver operating characteristic (roc) curves. *Canadian Journal of Emergency Medicine*, 8(1):19–20, 2006.

[14] C. Friedman, G. Hripcsak, W. DuMouchel, S. B. Johnson, and P. D. Clayton. Natural language processing in an operational clinical information system. *Natural Language Engineering*, 1(1):83–108, 1995.

[15] Carol Friedman, Lyudmila Shagina, Yves Lussier, and George Hripcsak. Automated Encoding of Clinical Documents Based on Natural Language Processing. *Journal of the American Medical Informatics Association*, 11(5):392–402, 09 2004.

[16] The Federal Government. Nhe fact sheet cms. 2022.

[17] RoseMary Hedberg. Analyzing lexical tool's fruitful variants for concept mapping in the synonym mapping tool. *NLM Associate Fellowship Projects, nlm.nih.gov*, 2013.

[18] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312, 2019.

[19] Chao-Wei Huang, Shang-Chi Tsai, and Yun-Nung Chen. PLM-ICD: Automatic ICD coding with pretrained language models. In Tristan Naumann, Steven Bethard, Kirk Roberts, and Anna Rumshisky, editors, *Proceedings of the 4th Clinical Natural Language Processing Workshop*, pages 10–20, Seattle, WA, July 2022. Association for Computational Linguistics.

[20] JohnSnowLabs. Spark nlp display. `https://github.com/JohnSnowLabs/spark-nlp-display/blob/main/assets/er_viz.png`, 2024.

[21] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv. *PhysioNet. Available online at: https://physionet. org/content/mimiciv/1.0/(accessed August 23, 2021)*, 2020.

[22] Alistair E. W. Johnson, Mohammad M. Ghassemi, Shamim Nemati, Katherine E. Niehaus, David A. Clifton, and Gari D. Clifford. Machine learning and decision support in critical care. *Proceedings of the IEEE*, 104(2):444–466, 2016.

[23] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[24] Veysel Kocaman and David Talby. Spark nlp: natural language understanding at scale. *Software Impacts*, 8:100058, 2021.

[25] Clemens Scott Kruse, Rishi Goswamy, Yesha Jayendrakumar Raval, and Sarah Marawi. Challenges and opportunities of big data in health care: a systematic review. *JMIR medical informatics*, 4(4):e5359, 2016.

[26] Leah S Larkey and W Bruce Croft. Automatic assignment of icd9 codes to discharge summaries. Technical report, Technical report, University of Massachusetts at Amherst, Amherst, MA, 1995.

[27] Daniel R Levinson, D Grant, J Durley, R Bessette, and M Verges. Improper payments for evaluation and management services cost medicare billions in 2010. *Department of Health and Human Services*, 2014.

[28] Fei Li and Hong Yu. Icd coding from clinical text using multi-filter residual convolutional neural network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8180–8187, Apr. 2020.

[29] Julia Medori and Cedrick Fairon. Machine learning and features selection for semi-automatic icd-9-cm encoding. In *Louhi '10: Proceedings of the NAACL HLT 2010 Second Louhi Workshop on Text and Data Mining of Health Documents*. Association for Computational Linguistics, 2010.

[30] Amnon Meyers. Multi-pass multi-strategy NLP. 2003.

[31] Stéphane M Meystre and Peter J Haug. Comparing natural language processing tools to extract medical problems from narrative text. In *AMIA annual symposium proceedings*, volume 2005, page 525. American Medical Informatics Association, 2005.

[32] Iwao Milton Moriyama, Ruth M Loy, Alastair Hamish Tearloch Robb-Smith, Harry Michael Rosenberg, and Donna L Hoyert. History of the statistical classification of diseases and causes of death. 2011.

[33] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. *arXiv preprint arXiv:1802.05695*, 2018.

[34] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[35] Travis B Murdoch and Allan S Detsky. The inevitable application of big data to health care. *Jama*, 309(13):1351–1352, 2013.

[36] Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2):124–130, 2010.

[37] Department of Health and Human Services. Health insurance reform: Standards for electronic transactions. Federal Register Volume 65, no. 160, 45 CFR Parts 160 and 162, August 17 2000.

[38] Department of Health and Human Services. Hipaa administrative simplification: Modifications to medical data code set standards to adopt icd–10–cm and icd–10–pcs. Federal Register Volume 74, no. 11, 45 CFR Part 162, January 16 2009.

[39] National Library of Medicine (US). Umls reference manual, September 2009. https://www.ncbi.nlm.nih.gov/books/NBK9676/.

[40] Kimberly J O'malley, Karon F Cook, Matt D Price, Kimberly Raiford Wildes, John F Hurdle, and Carol M Ashton. Measuring diagnoses: Icd code accuracy. *Health services research*, 40(5p2):1620–1639, 2005.

[41] Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, 21(2):231–237, 2014.

[42] Seth M Powsner. Automatic coding of medical problem lists. (3041), 1978.

[43] Guergana K Savova, Anni R Coden, Igor L Sominsky, Rie Johnson, Philip V Ogren, Piet C De Groen, and Christopher G Chute. Word sense disambiguation across two domains: biomedical literature and clinical notes. *Journal of biomedical informatics*, 41(6):1088–1100, 2008.

[44] Martijn J Schuemie, Jan A Kors, and Barend Mons. Word sense disambiguation in the biomedical domain: an overview. *Journal of Computational Biology*, 12(5):554–565, 2005.

[45] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[46] Thomas Searle, Zina Ibrahim, and Richard JB Dobson. Experimental evaluation and development of a silver-standard for the mimic-iii clinical coding dataset. *arXiv preprint arXiv:2006.07332*, 2020.

[47] DD Sherertz, MS Tuttle, NE Olson, MS Erlbaum, and SJ Nelson. Lexical mapping in the umls metathesaurus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 494. American Medical Informatics Association, 1989.

[48] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[49] Mary H Stanfill, Margaret Williams, Susan H Fenton, Robert A Jenders, and William R Hersh. A systematic literature review of automated clinical coding and classification systems. *Journal of the American Medical Informatics Association*, 17(6):646–651, 2010.

[50] Michael Q Stearns, Colin Price, Kent A Spackman, and Amy Y Wang. Snomed clinical terms: overview of the development process and project status. In *Proceedings of the AMIA Symposium*, page 662. American Medical Informatics Association, 2001.

[51] Ole Tange. Gnu parallel 20220722 ('roe vs wade'), July 2022. GNU Parallel is a general parallelizer to run multiple serial command line programs in parallel without changing them.

[52] Maxim Topaz, Leah Shafran-Topaz, and Kathryn H Bowles. Icd-9 to icd-10: evolution, revolution, and current debates in the united states. *Perspectives in Health Information Management/AHIMA, American Health Information Management Association*, 10(Spring), 2013.

[53] Thanh Vu, Dat Quoc Nguyen, and Anthony Nguyen. A label attention model for icd coding from clinical text. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3335–3341, 7 2020. Main track.

[54] Zhichao Yang, Shufan Wang, Bhanu Pratap Singh Rawat, Avijit Mitra, and Hong Yu. Knowledge injected prompt based fine-tuning for multi-label few-shot icd coding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2022, page 1767. NIH Public Access, 2022.

[55] Zheng Yuan, Chuanqi Tan, and Songfang Huang. Code synonyms do matter: Multiple synonyms matching network for automatic ICD coding. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 808–814, Dublin, Ireland, May 2022. Association for Computational Linguistics.