Low-resource ICD Coding of Hospital Discharge Summaries

Ashton Williamson[1], **David de Hilster**[2], **Amnon Meyers**[3], **Nina Hubig**[1], **and Amy Apon**[1]

[1]School of Computing, Clemson University
[2]LexisNexis Risk Solutions
[3]Conceptual Systems LLC
taw2@clemson.edu

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. 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 work we utilize the open-source NLP programming language, NLP++, 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 find that for codes with little training data, our approach achieves competitive performance compared to state-of-the-art machine learning approaches.

1 Introduction

Medical coding is the process by which healthcare 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 (Moriyama et al., 2011), and assessing quality of patient care (O'Malley et al., 2005). 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. We focus on the task of assigning International Classification of Diseases (ICD) codes to patient health records, which is a complex, multi-stage process with many possible points of failure, often resulting in improperly assigned or missing codes. 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 (Levinson et al., 2014). Medical coding thus presents itself as a critical task which would greatly benefit from increased automation.

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). Much research has been conducted in this area but it remains a challenging problem. Inherent limitations of state-of-the-art approaches tend to restrict their practical utility (Dong et al., 2022). Challenges include large label spaces and long document lengths, user requirements for explainability, and adaptability to local facility needs and medical advances. Classical machine-learning and deep-learning based approaches tend to be limited by the need for quality labeled data for supervised training. Due to restrictions on the distribution of patient medical data, collecting and curating useful datasets for these tasks is a major challenge (Johnson et al., 2016a; Searle et al., 2020; Johnson et al., 2016b). Annotation costs to develop gold-standard datasets can be prohibitive (Searle et al., 2020).

In this paper we propose a system for ICD coding that provides support for explainability and functions without training data. Our system first extracts medical entities from an input document, then maps these entities to concepts in the Unified Medical Language System (UMLS) (Bodenreider, 2004). Finally, we use these concepts as terms to assign ranking scores to ICD-9 codes for the input note. We use the openly available MIMIC-III dataset to evaluate the performance of our system and to compare to existing state-of-the-art ap-

Proceedings of the 23rd Workshop on Biomedical Language Processing, pages 548–558
August 16, 2024.

proaches to the task. Our contributions are to:

- Design and implement an automated end-to-end scalable system using readily available medical knowledge sources to assign ICD-9 codes to discharge summaries,
- Compare our approach to state-of-the-art deep learning approaches, and
- Demonstrate the utility of knowledge-based algorithms for low-resource ICD coding

2 Background

2.1 International Classification of Diseases

The ICD is a standardized nomenclature and classification system for diseases and medical procedures, which was originally intended to facilitate the statistical analysis of health data (Moriyama et al., 2011). Each successive revision to the ICD, typically spanning 10-20 years, has sought to address new use cases while adapting to advances in medicine and healthcare, and has continued to grow in number of total codes. The tenth version, ICD-10, has nearly 72,000 procedure codes. We utilize the ICD-9-CM, a clinical modification of the ICD-9 adapted for use in the US, which contains well over 10,000 codes. 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 (see Figure 1). Practical applications of the ICD in healthcare have expanded and now have come to include the indexing of health record data in hospitals, the coding of medical billing claims (Moriyama et al., 2011), and the assessment of quality of patient care (O'Malley et al., 2005).

2.2 MIMIC

To evaluate our approach to ICD coding we use the Medical Information Mart for Intensive Care (MIMIC) dataset (Johnson et al., 2016b). 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. There are four releases of the MIMIC dataset. Our work focuses on the third release as full access to the fourth release was not available until late in the project. The third release, MIMIC-III, was published in 2016 and contains data for 53,423 distinct hospital admissions. Each hospital admission record is comprehensive and includes data in one of the following categories: billing, descriptive, dictionary, interventions, laboratory, medications, notes, physiologic, and reports. MIMIC-III includes free text notes and reports such as radiology reports and hospital discharge summaries as well as ICD-9 codes for a hospital admission, making it a useful resource for the development and evaluation of automated code extraction.

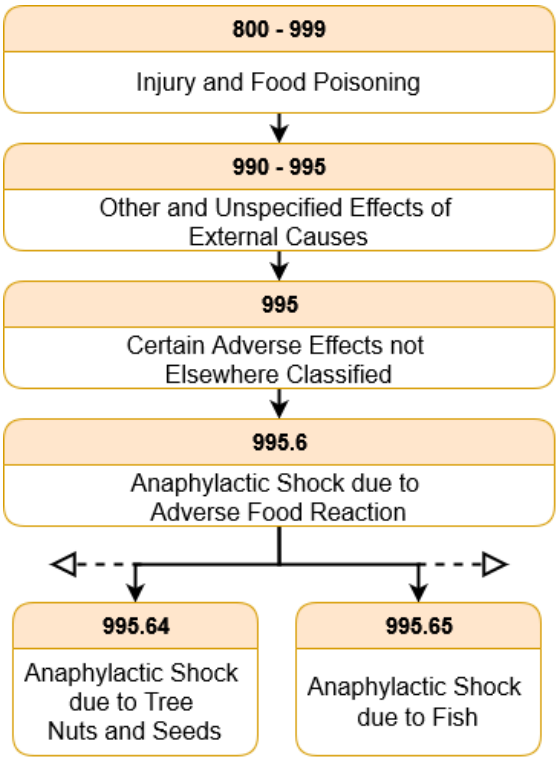

Figure 1: Example path in the ICD-9 hierarchy.

2.3 Unified Medical Language System

The UMLS is a repository of biomedical vocabularies and associated tools that is developed and maintained by the US National Library of Medicine. The UMLS consists of the Metathesaurus, a biomedical thesaurus that 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 (US National Library of Medicine, 2009). The Metathesaurus is a collection of source vocabularies including biomedical thesauri, classification systems, coding systems, and controlled term lists such as SNOMED-CT (Stearns et al., 2001). Terms in these vocabularies are linked to standard identifiers using semantic

or lexical information. The Semantic Network defines 127 different semantic types for concepts as well as 54 different relationships between them, comprising a network in which types are the nodes and relationships are the vertices. We leverage 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 medical terms that was developed with the express purpose of aiding in natural language processing of medical text (US National Library of Medicine, 2009). The lexicon includes key linguistic information for each term, including spelling variations, conjugations or conjugation patterns, plural forms, and more (Browne et al., 2000). 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 synonyms of terms within the ICD-9 code titles.

2.4 NLP++

We select the natural language processing programming language NLP++ to implement our system (Deane et al., 2001). NLP++ utilizes a *multi-pass, multi-strategy* architecture in which each user-defined 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, code regions include code which is executed at runtime, and declarative regions include user-defined functions that 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. Sequences of passes are grouped together as an *analyzer* tailored to a particular application.

NLP++ also incorporates a hierarchical knowledge base management system that allows the programmer to dynamically store and use information extracted from input texts. The conceptual grammar includes both knowledge bases and dictionaries. Knowledge bases allow the user to store and retrieve hierarchically structured information while dictionaries consist of entries and corresponding key/value pairs. After tokenization, a lookup is performed on the parse tree. Nodes that match dictionary entries are tagged with their respective key/value pairs. This facility constitutes a key aspect of building effective analyzers for parsing text.

3 Related Work

Though research on automated medical coding dates as least as far back as the 1970's (Powsner, 1978; Stanfill et al., 2010), access to data and hardware limitations prevented the development of large-scale solutions. The first work on ICD coding was published in the 1990s (Larkey and Croft, 1995). It treated the task as one of information retrieval, employing k-nearest-neighbors, relevance feedback, and Bayesian classifiers to select and rank relevant codes. Other early approaches leveraged biomedical entity recognition systems to extract clinically-significant entities which could then be linked to codes from the target coding system (Barrows Jr et al., 2000; Friedman et al., 2004). 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 (Medori and Fairon, 2010). 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. The approach separates the task into an extraction and classification step and inspires our approach to isolating relevant context which is then used for code classification.

Mullenbach et al. (Mullenbach et al., 2018) implement an attentional convolutional network to assign ICD-9 codes to discharge summaries in the MIMIC-III dataset. They introduce train, development and test splits for the Full set of MIMIC-III discharge summaries as well as a Top 50 split that includes only the 50 most frequently assigned codes. Both the Full and Top-50 splits defined by Mullenbach et al. have become the standard for comparison in the literature (Yang et al., 2022). We evaluate our system on the test set of these splits.

Yang et al. (Yang et al., 2022) 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 disease subset, MIMIC-III-

	Full	Top 50	Rare 50
Number of Notes	52,723	11,368	391
Number of Patients	41,126	10,356	386
Number of Unique Codes	8,922	50	50
Mean Codes per Note	15.9	5.7	1.0
Train/Dev/Test %	91/3/6	71/14/15	61/5/34

Table 1: Splits of the MIMIC-III dataset, including Full (Mullenbach et al., 2018), Top 50 (Mullenbach et al., 2018), and Rare 50 (Yang et al., 2022).

rare50, includes less-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 (Yang et al., 2022). To create this set, the authors first select codes with fewer than 10 occurrences, then select the top 50 from this set after splitting between train and test. Common diseases are also manually removed, resulting in 50 codes in total. 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 our system in a low-resource setting.

4 Methods

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 (Mullenbach et al., 2018). ICD9 codes are then generated from the PROCEDURES_ICD and DIAGNOSES_ICD tables using the subject and hospital admission IDs.

4.1 Domain Knowledge Integration

The UMLS serves two key roles in our system. The first is to identify clinically significant terms within ICD-9 titles. 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 (EUIs). These terms can be single words or n-grams within the title. For example, ICD-9 code 285.1 with title *Acute posthemorrhagic anemia*, generates EUIs E0007202, *acute posthemmorhagic anemia*; E0049207, *posthemmorhagic*; E0007127, *acute*; and E0008920, *anemia* (Figure 2, Step 1).

In the second step, we leverage the normalized terms identified in the first step to generate sets of alternative forms of these terms (see Step 2 of Figure 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 (Sherertz et al., 1989). 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. We utilize the fruitful variants flag, which includes spelling variants, inflections, synonyms, acronyms and abbreviations, expansions of abbreviations and acronyms, and derivations (Divita et al., 2014). This information is aggregated and organized into knowledge bases and dictionaries for downstream use in our analyzer.

4.2 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. 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 and link these to central concepts. In the third step we rank the set of extracted codes using an inverse-document frequency-based method. In this section we give an overview of the analyzer structure and experimental setup to inves-

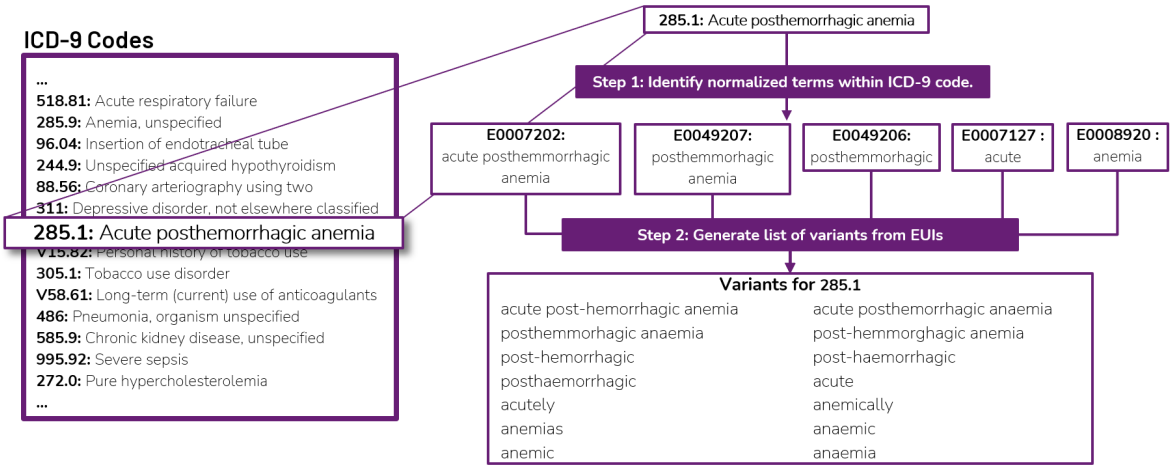

Figure 2: Outline of the pipeline for term normalization and variant generation for ICD-9 titles.

tigate 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. 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. The output of the tokenization step is 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 (i.e., *KBFuncs*, *array_funcs*, and *pn_funcs*) are declarative passes that define functions which are called throughout the analyzer. *KBFuncs* is a library pass that provides useful functions for working with knowledge bases. We utilize NLP++ built-in functions to add unique strings, concepts, and values to knowledge bases 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 that appends a value to a parse tree node's variable.

Passes 6 through 19 perform cleaning of the note text and organization of the parse tree. Starting with pass 6 we excise non-relevant information including de-identified placeholder strings, headers and footers, which empirical investigation suggests predominantly contain metadata. After filtering, we organize the parse tree into structural components in order of increasing granularity: sections, subsections, enumerated lists, and sentences. For sections and subsections, header names (e.g., "History of Patient Illness" or "Chief Complaint") 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 (Chapman et al., 2001) for its relative effectiveness and ease of implementation, though some work has shown improvement using a dynamic window size (Meystre and Haug, 2005). The first rule in the pass matches a leaf node tagged as *pre*-negation, 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 next rule performs the same operation for *post*-negation terms, instead excising the preceding 5 nodes.

Passes 21-27 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 (Sparck Jones, 1972). The set of all ICD-9 titles is the reference corpus, *C*. 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 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) (Sparck Jones, 1972). 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, or $\mathcal{W}_c = \sum_i^{|T_c|} \frac{f_{t_i}}{|C|}$. 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:

$$\mathcal{R}_c = \frac{\sum_{j=1}^{|H|} \mathcal{W}_{H_j}}{\mathcal{W}_c} \quad (1)$$

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.

4.3 Evaluation

We evaluate all approaches on the test sets of the Top 50 and Rare 50 splits of MIMIC-III (for a comparison of these splits, see Table 1). Following previous work (Mullenbach et al., 2018; Yang et al., 2022), we use the receiver operating characteristic area under the curve (*ROC AUC*), F1-score and 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.

4.4 Execution Characteristics

The note processing stage is conducted in parallel on the Clemson Palmetto HPC cluster using only CPUs. Notes are first written to individual text files, which are then mapped to available processes with GNU Parallel (Tange, 2022). Each process runs an instance of the note processing analyzer 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.

5 Results

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.

5.1 Rare 50

Results for the Rare 50 split are shown in Table 3. We find that our LexSyn-Section$_{\text{max}}$ analyzer achieves a level of performance comparable to recent state-of-the-art approaches in terms of ROC AUC, falling within 4 points of KEPTLongformer$_{\textit{finetuned}}$, the best-performing deep-learning model.

Despite the slight performance difference, we identify a few key advantages which support the

Approach	ROC AUC		F1-Score		Prec. @ k
	Micro	Macro	Micro	Macro	5
CAML (Mullenbach et al., 2018)	91.1	87.5	52.4	60.6	61.1
PLM-ICD (Huang et al., 2022)	91.9	89.3	67.6	64.3	61.7
MSMN (Yuan et al., 2022)	94.7	92.8	72.2	68.1	67.6
KEPTLongformer (Yang et al., 2022)	94.2	92.0	72.7	68.5	67.4
LexSyn-Section$_{max}$	69.8	70.9	33.3	37.1	29.4
LexSyn-Section$_{mean}$	67.8	68.9	30.1	33.7	26.3
LexSyn-Section$_{sum}$	68.6	71.3	31.7	39.6	27.6
LexSyn-Sent$_{max}$	69.6	69.1	31.3	34.4	29.0
LexSyn-Sent$_{mean}$	**71.7**	72.2	**34.5**	37.1	**31.8**
LexSyn-Sent$_{sum}$	70.0	**72.2**	32.9	**40.8**	27.7
LexSyn-Full	68.0	68.0	31.9	38.5	29.6

Table 2: Results on the MIMIC-III Top 50 test set (Mullenbach et al., 2018). Results for all approaches are run to completion. The best performing result for each metric from our approaches is bolded.

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

Table 3: **MIMIC-III Rare 50 test set results**. Results for previous approaches from (Yang et al., 2022). The best performing result for each metric from our approaches is bolded.

utility of our analyzer in a clinical setting. The first of these is the potential for explainability, as described in the next section. 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.

5.2 Top 50

Results for the Top 50 split are shown in Table 2. Results on the Top 50 split are comparable to the results for the Rare 50 split but not as close to recent state-of-the-art deep learning methods on the Top 50 split. We note that the Top 50 split has a smaller label space (50 labels vs 8,922 for the full set) and a large number of samples per label, making this dataset significantly less challenging for deep learning methods than the Rare 50 and Full sets. We nonetheless find that our approaches achieve a reasonable performance baseline.

We perform additional analysis on individual codes to explain the resulting code predictions. Observation of the per-code F1-scores is shown in Figure 3. Performance for individual codes on the Top 50 dataset is highly variable, with F1 scores that range from nearly 0.9 to 0.0. We conduct analysis for the three ICD-9 codes with individual F1 scores equal to 0.0 (*412*, *285.1*, and *39.61*). We plot the confusion matrices, seen in Figure 4, for these codes and observe for these codes a positive label is never or almost never predicted.

For code *412*, "Old myocardial infarction", manual inspection reveals that this code 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 significantly outperforms our approach, with an F1 score on the Top 50 test set of 58.2. Although our system does not leverage code title information to restrict code matches to sections in the text, our system does provide support for incorporating this type of rule.

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

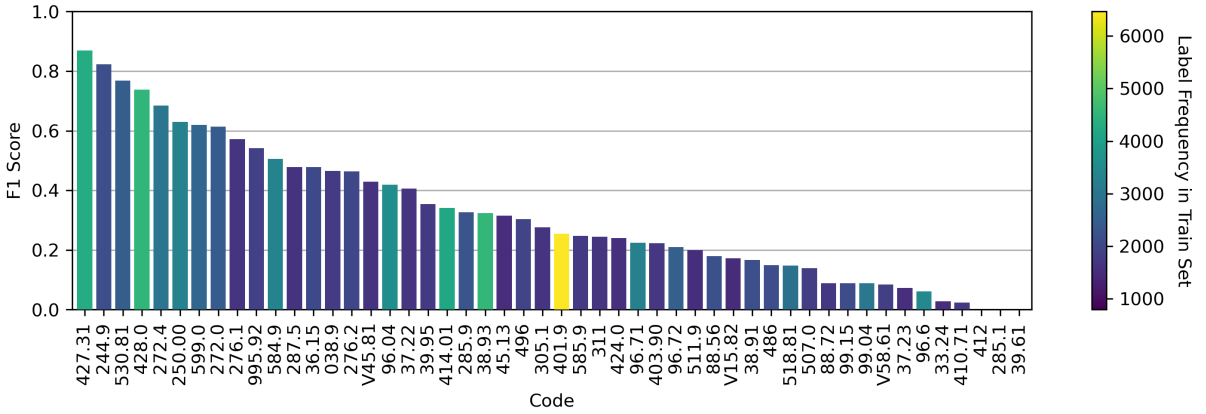

Figure 3: F1-score per code on the Top-50 dataset, sorted by decreasing F1-score. Bar colors represent frequency of the code in the Top-50 training set.

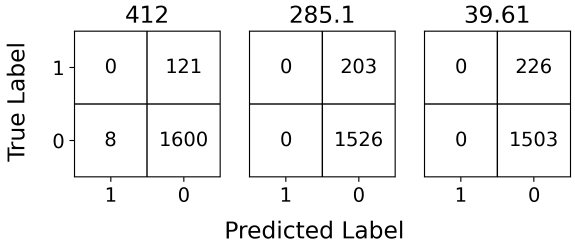

Figure 4: Confusion matrices for the three codes in the Top 50 set with F1-Score equal to 0.0.

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 current 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 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 helpful for improving performance.

5.3 Codes with Multiple Occurrences

For the Top 50 and Rare 50 datasets, our analyzer generates code ranks for each sentence or section. When the same code occurs in multiple sentences or sections the result is a large number of ranking scores for a code in the note. To handle the situation in which a code occurs multiple times in the same note, we use one of three aggregation methods: the mean, the sum or the median of the ranks. However, we suspect that noisy ranking scores for frequent terms adversely affect performance.

6 Future Work

Our approach to extracting clinical entities 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 (Savova et al., 2008). Consider, for example, the term 'ms', which maps to 12 unique concepts in the 2007AC UMLS (Savova et al., 2008). 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) (Schuemie et al., 2005; Chasin et al., 2014) into the entity extraction passes of the analyzer.

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 4.1. We find the literature on lexical normalization for medical entities to be sparse (Divita et al., 2014; Hedberg, 2013). 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.

The dataset on which we evaluate our system, MIMIC-III (Johnson et al., 2016b), 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 (Searle et al., 2020). As our system is developed and built entirely from available medical sources using knowledge-based algorithms, the system output is consistent and reproducible. It is potentially usable as a companion to deep learning based methods to aid in the development of gold-standard labels for the MIMIC dataset.

Perhaps the most critical area for future research that we identify is the potential of our approach to be utilized as an assistive software for human coders. In this role our system does not replace human coders but can be used as a "first pass" to coding or to flag inconsistencies for human verification. Code for our system is made available at `https://github.com/ashtonomy/low_resource_icd_coding`.

Limitations

Despite showing competitive performance in few- or zero-shot settings, our analyzer is limited by its performance in high-resource settings, such as the Top 50 test set discussed in Section 5. More work is needed to improve performance in this domain before deployment in a clinical setting is considered. We also note that our system is evaluated on medical notes sourced from a single hospital system. In general, we find that more robust evaluation on data from different source domains is needed to more effectively gauge performance. As discussed in 6, this is a challenge at present due to limited access to openly available annotated medical notes.

Ethics Statement

In this natural language processing research we adhere to the highest ethical standards to ensure integrity, transparency, and respect for all contributors and subjects involved. We recognize the potential impacts of NLP technologies on society and are committed to responsible research practices. This version of the system is a proof-of-concept and is intended for research purposes only. It has not been validated for production use.

Acknowledgments

The authors acknowledge the generous support of HPCC Systems and LexisNexisRisk. This material is based on work supported by the National Science Foundation under Grant Nos. MRI# 2024205, MRI# 1725573, and CRI# 2010270. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Clemson University is acknowledged for their generous allotment of compute time on the Palmetto Cluster.

References

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

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

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

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

Rachel Chasin, Anna Rumshisky, Ozlem Uzuner, and Peter Szolovits. 2014. 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.

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

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

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

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

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

Chao-Wei Huang, Shang-Chi Tsai, and Yun-Nung Chen. 2022. PLM-ICD: Automatic ICD coding with pretrained language models. In *Proceedings of the 4th Clinical Natural Language Processing Workshop*, pages 10–20, Seattle, WA. Association for Computational Linguistics.

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

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. 2016b. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

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

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

Julia Medori and Cedrick Fairon. 2010. 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.

Stéphane M Meystre and Peter J Haug. 2005. 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.

Iwao Milton Moriyama, Ruth M Loy, Alastair Hamish Tearloch Robb-Smith, Harry Michael Rosenberg, and Donna L Hoyert. 2011. History of the statistical classification of diseases and causes of death. *DHHS publication*, 2011-1125.

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

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

Seth M Powsner. 1978. Automatic coding of medical problem lists. *Yale Medicine Thesis Digital Library*, (3041).

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

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

Thomas Searle, Zina Ibrahim, and Richard Dobson. 2020. Experimental evaluation and development of a silver-standard for the MIMIC-III clinical coding dataset. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 76–85, Online. Association for Computational Linguistics.

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

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

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

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

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

US National Library of Medicine. 2009. Umls reference manual. Https://www.ncbi.nlm.nih.gov/books/NBK9676/.

Zhichao Yang, Shufan Wang, Bhanu Pratap Singh Rawat, Avijit Mitra, and Hong Yu. 2022. 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.

Zheng Yuan, Chuanqi Tan, and Songfang Huang. 2022. Code synonyms do matter: Multiple synonyms matching network for automatic icd coding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 808–814.

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